BIG-IP® Controller Reference Guide

version 3.3

Service and Support Information

Product Version

This manual applies to version 3.3 of the BIG-IP® Local Traffic Controller.

Obtaining Technical Support

Web	tech.f5.com
Phone	(206) 272-6888
Fax	(206) 272-6802
Email (support issues)	support@f5.com
Email (suggestions)	feedback@f5.com

Contacting F5 Networks

Web	www.f5.com
Toll-free phone	(888) 961-7242
Corporate phone	(206) 272-5555
Fax	(206) 272-5556
Email	sales@f5.com
Mailing Address	501 Elliott Avenue West Seattle, Washington 98119

Legal Notices

Copyright

F5 Networks, Inc. (F5) believes the information it furnishes to be accurate and reliable. However, F5 assumes no responsibility for the use of this information, nor any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent, copyright, or other intellectual property right of F5 except as specifically described herein. F5 reserves the right to change specifications at any time without notice.

Copyright 1997-2000, F5 Networks, Inc. All rights reserved.

Trademarks

F5, BIG-IP, and 3-DNS are registered trademarks of F5 Networks, Inc. SEE-IT, GLOBAL-SITE, EDGE-FX, and FireGuard are trademarks of F5 Networks, Inc. Other product and company names are registered trademarks or trademarks of their respective holders.

Export Regulation Notice

The BIG-IP® Local Traffic Controller may include cryptographic software. Under the Export Administration Act, the United States government may consider it a criminal offense to export this BIG-IP® Local Traffic Controller from the United States.

Export Warning

This is a Class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

FCC Compliance

This equipment generates, uses, and may emit radio frequency energy. The equipment has been type tested and found to comply with the limits for a Class A digital device pursuant to Part 15 of FCC rules, which are designed to provide reasonable protection against such radio frequency interference.

Operation of this equipment in a residential area may cause interference, in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Any modifications to this device, unless expressly approved by the manufacturer, can void the user's authority to operate this equipment under part 15 of the FCC rules.

Canadian Regulatory Compliance

This class A digital apparatus complies with Canadian I CES-003.

Standards Compliance

The product conforms to ANSI/UL Std 1950 and Certified to CAN/CSA Std. C22.2 No. 950.

Acknowledgments

This product includes software developed by the University of California, Berkeley and its contributors.

This product includes software developed by the Computer Systems Engineering Group at the Lawrence Berkeley Laboratory.

This product includes software developed by the NetBSD Foundation, Inc. and its contributors.

This product includes software developed by Christopher G. Demetriou for the NetBSD Project.

This product includes software developed by Adam Glass.

This product includes software developed by Christian E. Hopps.

This product includes software developed by Dean Huxley.

This product includes software developed by John Kohl.

This product includes software developed by Paul Kranenburg.

This product includes software developed by Terrence R. Lambert.

This product includes software developed by Philip A. Nelson.

This product includes software developed by Herb Peyerl.

This product includes software developed by Jochen Pohl for the NetBSD Project.

This product includes software developed by Chris Provenzano.

This product includes software developed by Theo de Raadt.

This product includes software developed by David Muir Sharnoff.

This product includes software developed by SigmaSoft, Th. Lockert.

This product includes software developed for the NetBSD Project by Jason R. Thorpe.

This product includes software developed by Jason R. Thorpe for And Communications, http://www.and.com.

This product includes software developed for the NetBSD Project by Frank Van der Linden.

This product includes software developed for the NetBSD Project by John M. Vinopal.

This product includes software developed by Christos Zoulas.

This product includes software developed by Charles Hannum.

This product includes software developed by Charles Hannum, by the University of Vermont and State Agricultural College and Garrett A. Wollman, by William F. Jolitz, and by the University of California, Berkeley, Lawrence Berkeley Laboratory, and its contributors.

This product includes software developed by the University of Vermont and State Agricultural College and Garrett A. Wollman.

In the following statement, "This software" refers to the Mitsumi CD-ROM driver: This software was developed by Holger Veit and Brian Moore for use with "386BSD" and similar operating systems. "Similar operating systems" includes mainly non-profit oriented systems for research and education, including but not restricted to "NetBSD," "FreeBSD," "Mach" (by CMU).

In the following statement, "This software" refers to the parallel port driver: This software is a component of "386BSD" developed by William F. Jolitz, TeleMuse.

This product includes software developed by the Apache Group for use in the Apache HTTP server project

(http://www.apache.org/).

This product includes software developed by Darren Reed. (© 1993-1998 by Darren Reed).

This product includes software licensed from Richard H. Porter under the GNU Library General Public License (© 1998, Red Hat Software), www.gnu.org/copyleft/lgpl.html.

This product includes the standard version of Perl software licensed under the Perl Artistic License (© 1997, 1998 Tom Christiansen and Nathan Torkington). All rights reserved. You may find the most current standard version of Perl at http://www.perl.com.

F5 Networks Limited Warranty

This warranty will apply to any sale of goods or services or license of software (collectively, "Products") from F5 Networks, Inc. ("F5"). Any additional or different terms including terms in any purchase order or order confirmation will have no effect unless expressly agreed to in writing by F5. Any software provided to a Customer is subject to the terms of the End User License Agreement delivered with the Product.

Limited Warranty

Software. F5 warrants that for a period of 90 days from the date of shipment: (a) the media on which the software is furnished will be free of defects in materials and workmanship under normal use; and (b) the software substantially conforms to its published specifications. Except for the foregoing, the software is provided AS IS.

In no event does F5 warrant that the Software is error free, that the Product will operate with any software or hardware other than that provided by F5 or specified in the documentation, or that the Product will satisfy Purchaser's own specific requirements.

Hardware. F5 warrants that the hardware component of any Product will, for a period of one year from the date of shipment from F5, be free from defects in material and workmanship under normal use.

Remedy. Purchaser's exclusive remedy and the entire liability of F5 under this limited warranty and any other guarantee made by F5 is, at F5's option, to repair or replace any Product or component that fails during the warranty period at no cost to Purchaser. Products returned to F5 must be pre-authorized by F5 with a Return Material Authorization (RMA) number marked on the outside of the package, and sent prepaid, insured, and packaged appropriately for safe shipment. The repaired or replaced item will be shipped to Purchaser, at F5's expense, no later than 7 days after receipt by F5. Title to any returned Products or components will transfer to F5 upon receipt. F5 will replace defective media or documentation or, at its option, undertake reasonable efforts to modify the software to correct any substantial non-conformance with the specifications.

Restrictions. The foregoing limited warranties extend only to the original Purchaser, and do not apply if a Product (a) has been altered, except by F5, (b) has not been installed, operated, repaired, or maintained in accordance with F5's instructions, (c) has been subjected to abnormal physical or electrical stress, misuse, negligence, or accident, or (d) has been operated outside of the environmental specifications for the Product. F5's limited software warranty does not apply to software corrections or upgrades.

Support, Upgrades. F5 provides software telephone support services at no charge for 90 days following the installation of any Product: Monday through Friday, from 6 a.m. to 6 p.m. Pacific time, excluding F5's holidays. Such support will consist of responding to trouble calls as reasonably required to make the Product perform as described in the Specifications. For advisory help requests, which are calls of a more consultative nature than a standard trouble call, F5 will provide up to two hours of telephone service at no charge. Additional service for advisory help requests may be purchased at F5 Networks' then-current standard service fee. During this initial 90

day period, Customer is entitled, at no charge, to updated versions of covered software such as bug fixes, and incremental enhancements as designated by minor revision increases. In addition, Customer will receive special pricing on upgraded versions of covered Products such as new clients, new modules, and major enhancements designated by major revision increases. Customer may purchase a Maintenance Agreement for enhanced maintenance and support services.

DISCLAIMER; LIMITATION OF REMEDY: EXCEPT FOR THE WARRANTIES SPECIFICALLY DESCRIBED HEREIN, F5 DOES NOT MAKE ANY GUARANTEE OR WARRANTY, EXPRESS OR IMPLIED, WITH RESPECT TO PRODUCTS, SPECIFICATIONS, SUPPORT, SERVICE, OR ANYTHING ELSE. F5 HAS NOT AUTHORIZED ANYONE TO MAKE ANY REPRESENTATION OR WARRANTY OTHER THAN AS PROVIDED ABOVE. F5 DISCLAIMS ANY AND ALL WARRANTIES AND GUARANTEES, EXPRESS, IMPLIED, OR OTHERWISE, ARISING WITH RESPECT TO THE PRODUCTS OR SERVICES DELIVERED HEREUNDER, INCLUDING BUT NOT LIMITED TO THE WARRANTY OF MERCHANTABILITY, THE WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE, AND ANY WARRANTY OF NON-INFRINGEMENT OF THE INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY. F5 WILL HAVE NO OBLIGATION OR LIABILITY, WHETHER ARISING IN CONTRACT (INCLUDING WARRANTY), TORT (INCLUDING ACTIVE, PASSIVE, OR IMPUTED NEGLIGENCE, STRICT LIABILITY, OR PRODUCT LIABILITY), OR OTHERWISE FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR INDIRECT DAMAGES INCLUDING BUT NOT LIMITED TO LOSS OF USE, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF REVENUE, LOSS OF BUSINESS, OR OTHER FINANCIAL LOSS ARISING OUT OF OR IN CONNECTION WITH ANY OF THE PRODUCTS OR OTHER GOODS OR SERVICES FURNISHED TO CUSTOMER BY F5, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

End-user Software License

IMPORTANT - READ BEFORE INSTALLING OR OPERATING THIS PRODUCT

CAREFULLY READ THE TERMS AND CONDITIONS OF THIS LICENSE BEFORE INSTALLING OR OPERATING THIS PRODUCT - BY INSTALLING, OPERATING OR KEEPING THIS PRODUCT FOR MORE THAN THIRTY DAYS AFTER DELIVERY YOU INDICATE YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS. IF YOU DO NOT AGREE WITH THE TERMS AND CONDITIONS OF THIS AGREEMENT, PROMPTLY CONTACT F5 NETWORKS, INC. ("F5") TO ARRANGE FOR RETURN OF THE PRODUCT FOR A REFUND.

1. Scope. This License applies to the software component ("Software") of the F5 product identified above ("Product") and any corrections, updates, new releases and new versions of such software. This License is a legal agreement between F5 and the single entity ("Licensee") that has acquired Software from F5 under applicable terms and conditions.

2. License Grant. Subject to the terms of this License, F5 grants to Licensee a non-exclusive, non-transferable license to use the Software in object code form with an unlimited number of servers. Other than as specifically described herein, no right or license is granted to Licensee to any of F5's trademarks, copyrights, or other intellectual property rights. The Software incorporates certain third party software, which is used subject to licenses from the respective owners. The protections given to F5 under this License also apply to the suppliers of this third party software, who are intended third party beneficiaries of this License.

3. Restrictions. The Software, documentation, and the associated copyrights and other intellectual

property rights are owned by F5 or its licensors, and are protected by law and international treaties. Licensee may not copy or reproduce the Software, and may not copy or translate the written materials without F5's prior, written consent. Licensee may not copy, modify, reverse compile, or reverse engineer the Software, or sell, sub-license, rent, or transfer the Software or any associated documentation to any third party.

4. Export Control. F5's standard Software incorporates cryptographic software. Licensee agrees to comply with the Export Administration Act, the Export Control Act, all regulations promulgated under such Acts, and all other US government regulations relating to the export of technical data and equipment and products produced therefrom, which are applicable to Licensee. In countries other than the US, Licensee agrees to comply with the local regulations regarding importing, exporting, or using cryptographic software.

5. Limited Warranty. F5 warrants that for a period of 90 days from the date of shipment: (i) the media on which the Software is furnished will be free of defects in materials and workmanship under normal use; and (ii) the Software substantially conforms to its published specifications. Except for the foregoing, the Software is provided AS IS. In no event does F5 warrant that the Software is error free, that it will operate with any software or hardware other than that provided by F5 or specified in the documentation, or that the Software will satisfy Licensee's own specific requirements.

   a. Remedy. Licensee's exclusive remedy and the entire liability of F5 under this limited warranty and any other guarantee made by F5 is, at F5's option, to repair or replace any Software that fails during the warranty period at no cost to Licensee. Any Product returned to F5 must be pre-authorized by F5 with a Return Material Authorization (RMA) number marked on the outside of the package, and sent prepaid, insured and packaged appropriately for safe shipment. The repaired or replaced item will be shipped to Licensee, at F5's expense, no later than 7 days after receipt by F5. Title to any returned Products or components will transfer to F5 upon receipt. F5 will replace defective media or documentation or, at its option, undertake reasonable efforts to modify the Software to correct any substantial non-conformance with the specifications.

   b. Restrictions. The foregoing limited warranties extend only to the original Licensee, and do not apply if the Software or the Product (a) has been altered, except by F5, (b) has not been installed, operated, repaired, or maintained in accordance with F5's instructions, (c) has been subjected to abnormal physical or electrical stress, misuse, negligence or accident or (d) has been operated outside of the environmental specifications for the Product. F5's limited software warranty does not apply to software corrections or upgrades.

6. Infringement Indemnity. F5 will, at its expense, defend any suit brought against Licensee based upon a claim that the Software as delivered by F5 directly infringes a valid patent or copyright. F5 will pay costs and damages finally awarded against Licensee directly attributable to any such claim, but only on condition that (a) F5 is notified in writing of such claim within ten days following receipt by Licensee; (b) F5 has sole control of the defense and settlement negotiations, (c) Licensee provides F5 all information and communications received by Licensee concerning such claim, and (d) Licensee provides reasonable assistance to F5 when requested. F5 will have the right, at its option and expense, (i) to obtain for Licensee rights to use the Software, (ii) to replace or modify the Software so it becomes non-infringing , or (iii) to accept return of the Software in exchange or for a credit not to exceed the purchase price paid by Licensee for such Software. The foregoing, subject to the following restrictions, states the exclusive liability of F5 to Licensee concerning infringement.

   a. Restrictions. F5 will have no liability for any claim of infringement based on: (i) use of a superseded or altered release of the Software, (ii) use of the Software in combination with equipment or software

not supplied or specified by F5 in the Software documentation where the Software would not itself be infringing, (iii) use of the Software in an application or environment not described in the Software Documentation or (iv) Software that has been altered or modified in any way by anyone other than F5 or according to F5's instructions.

7. U.S. Government Restricted Rights. The Software was developed at private expense and is provided with "RESTRICTED RIGHTS." Use, duplication or disclosure by the government is subject to restrictions as set forth in FAR 52.227-14 and DFARS 252.227-7013 et. seq. or its successor. The use of this Software by the government constitutes acknowledgment of F5's and its licensors' rights in the Software.

8. DISCLAIMER; LIMITATION OF REMEDY. EXCEPT FOR THE WARRANTIES SPECIFICALLY DESCRIBED HEREIN, F5 AND ITS THIRD PARTY LICENSORS DO NOT MAKE ANY GUARANTEE OR WARRANTY, EXPRESS OR IMPLIED, WITH RESPECT TO THE SOFTWARE, SPECIFICATIONS, SUPPORT, SERVICE OR ANYTHING ELSE. NEITHER F5 NOR ITS THIRD PARTY LICENSORS HAVE AUTHORIZED ANYONE TO MAKE ANY REPRESENTATIONS OR WARRANTIES OTHER THAN AS PROVIDED ABOVE. F5 AND ITS THIRD PARTY LICENSORS DISCLAIM ANY AND ALL WARRANTIES AND GUARANTEES, EXPRESS, IMPLIED OR OTHERWISE, ARISING, WITH RESPECT TO THE SOFTWARE OR SERVICES DELIVERED HEREUNDER, INCLUDING BUT NOT LIMITED TO THE WARRANTY OF MERCHANTABILITY, THE WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE, AND ANY WARRANTY OF NON-INFRINGEMENT OF THE INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY. THE COLLECTIVE LIABILITY OF F5 AND ITS THIRD PARTY LICENSORS UNDER THIS LICENSE WILL BE LIMITED TO THE AMOUNT PAID FOR THE PRODUCT. F5 AND ITS THIRD PARTY LICENSORS WILL HAVE NO OBLIGATION OR LIABILITY, WHETHER ARISING IN CONTRACT (INCLUDING WARRANTY), TORT (INCLUDING ACTIVE, PASSIVE OR IMPUTED NEGLIGENCE, STRICT LIABILITY OR PRODUCT LIABILITY) OR OTHERWISE FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES INCLUDING BUT NOT LIMITED TO LOSS OF USE, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF REVENUE, LOSS OF BUSINESS OR OTHER FINANCIAL LOSS ARISING OUT OF OR IN CONNECTION WITH THE SOFTWARE OR OTHER GOODS OR SERVICES FURNISHED TO LICENSEE BY F5, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

9. Termination. The license granted in Section 2 is effective until terminated, and will automatically terminate if Licensee fails to comply with any of its provisions. Upon termination, Licensee will destroy the Software and documentation and all copies or portions thereof.

10. Miscellaneous. This Agreement will be governed by the laws of the State of Washington, USA without regard to its choice of law rules. The provisions of the U.N. Convention for the International Sale of Goods will not apply. Any provisions found to be unenforceable will not affect the enforceability of the other provisions contained herein, but will instead be replaced with a provision as similar in meaning to the original as possible. This Agreement constitutes the entire agreement between the parties with regard to its subject matter. No modification will be binding unless in writing and signed by the parties.

F5 Networks BIG-IP® Local Traffic Controller

Table of Contents

Introduction

Getting started I-1
Choosing a configuration tool I-1
Using the Administrator Kit I-2
Stylistic conventions I-3
Finding additional help and technical support resources I-5
What's new in version 3.3 . . . . . I-6
BIG-IP e-Commerce Controller I-6
BIG-IP Cache Load Balancer I-6
Performance enhancements I-7
Learning more about the BIG-IP Controller product family I-7

Chapter 1

Object Properties

Object properties 1-1
Extended Application Verification (EAV) 1-2
Setting up custom EAV service checks 1-3
Using the EAV pingers bundled with the BIG-IP Controller . . . 1-7
Extended Content Verification (ECV) 1-13
ECV service check properties 1-15
Writing regular expressions for ECV service checks 1-16
Setting up ECV service checks in the Configuration utility . . . 1-18
Setting up ECV service checks for transparent nodes 1-19
Manually configuring and testing the /etc/bigd.conf file 1-22
Filters 1-25
IP filters 1-25
Rate filters and rate classes 1-27
IP forwarding 1-30
Setting up IP forwarding 1-30
Configuring forwarding virtual servers 1-32
Interface 1-34
Versatile interface configuration 1-35
Destination route and translation processing 1-37
Source translation processing 1-37
Source translation 1-39
Interface security 1-39
Displaying status for interfaces 1-40
Arming and disarming the fail-safe mode 1-41

Setting the fail-safe timeout 1-42
Viewing the timeout setting 1-42
Setting the MAC masquerade address 1-42
Enabling VLAN tags for an interface 1-44
Load Balancing 1-45
Understanding individual load balancing modes 1-46
Setting the global load balancing mode 1-48
Setting ratio weights and priority levels for node addresses . . . 1-49
Setting the load balancing method for a pool 1-51
NAT 1-53
Defining a network address translation (NAT) 1-55
Node 1-58
Adding a node as a member to a pool 1-61
Pool 1-62
Setting up persistence for a pool 1-65
Using HTTP cookie persistence 1-69
Using destination address affinity (sticky persistence) 1-75
Using a simple timeout and a persist mask on a pool 1-77
Maintaining persistence across virtual servers that use the same virtual addresses 1-78
Maintaining persistence across all virtual servers 1-79
Backward compatible persistence for node list virtual servers 1-81
Port 1-84
Redundant System 1-87
Preparing to use the synchronization command 1-88
Synchronizing configurations between controllers 1-89
Configuring fail-safe settings 1-91
Mirroring connection and persistence information 1-93
Using gateway fail-safe 1-96
Using network-based fail-over 1-98
Setting a specific BIG-IP Controller to be the preferred active unit 1-100
Setting up active-active redundant controllers 1-101
Configuring an active-active system 1-101
Active-active system fail-over 1-108
Additional active-active BIG/db configuration parameters . . 1-110
Specific active-active bigpipe commands 1-111
Running mixed versions of BIG-IP Controller software in active-active mode 1-112

Returning an active-active installation to active/standby mode 1-113
Rule 1-114
Pool selection based on HTTP request data 1-114
Pool selection based on IP packet header information 1-115
HTTP request string variables 1-119
Configuring rules 1-120
Configuring virtual servers that reference rules 1-121
Cache statement syntax 1-124
Additional rule examples 1-127
Comparing load balancing configurations 1-131
SNAT 1-133
timer settings 1-138
Setting the node ping timer 1-139
Setting the timer for reaping idle connections 1-141
Setting the service check timer 1-143
Service checking for wildcard servers and ports 1-144
Virtual server 1-146
Using standard or wildcard virtual servers 1-147
Defining virtual servers 1-147
Defining wildcard virtual servers 1-149
Configuring a network virtual server 1-153
Using additional features with virtual servers 1-161

Chapter 2
bigpipe Command Reference

bigpipe commands 2-1
-? 2-4
alias 2-5
configsync 2-7
conn 2-8
-d 2-9
-f 2-10
failover 2-11
gateway 2-12
-h and -help 2-13
interface 2-14
ipalias 2-21

-l 2-22
lb 2-23
maint 2-25
mirror 2-26
nat 2-27
-n 2-30
node 2-31
persist 2-35
pool 2-36
port 2-44
proxy 2-46
-r 2-50
ratio 2-51
rule 2-53
-s 2-57
snat 2-58
summary 2-62
timeout_node 2-64
timeout_svc 2-66
tping_node 2-68
tping_svc 2-69
treaper 2-71
udp 2-73
unit 2-75
-v 2-76
version 2-77
vip 2-78
Backward-compatible commands 2-87

Chapter 3

Configuration Files

Configuration files for the BIG-IP Controller 3-1

Chapter 4

BIG-IP Controller Configuration Utilities

Introducing the BIG-IP Controller configuration utilities 4-1
config 4-2

config_failover 4-4
reconfig-httpd 4-5
config_telnetd 4-6
config_sshd 4-7
config_rshd 4-8

Chapter 5

BIG-IP System Control Variables

Setting BIG-IP system control variables 5-1
sysctl 5-2
bigip.bonfire_mode 5-3
bigip.bonfire_compatibility_mode 5-4
bigip.fastest_max_idle_time 5-5
bigip.fastpath_active 5-6
bigip.forwarding_vip_overrides_default_snat 5-8
bigip.halt_reboot_timeout 5-9
bigip.improved_fastest 5-10
bigip.max_sticky_entries 5-11
bigip.memory_reboot_percent 5-12
bigip.open_3dns_lockdown_ports 5-13
bigip.open_telnet_port 5-14
bigip.open_ftp_ports 5-15
bigip.open_ssh_port 5-16
bigip.open_rsh_ports 5-17
bigip.persist_map_proxies 5-18
bigip.persist_time_used_as_limit 5-19
bigip.persist_on_any_vip 5-20
bigip.persist_on_any_port_same_vip 5-21
bigip.tcphps_mss_override 5-22
bigip.verbose_log_level 5-23
bigip.vipnoarp 5-24
bigip.webadmin_port 5-25
net.inet.ip.forwarding 5-26
net.inet.ip.sourcecheck 5-27

Chapter 6
System Utilities

Introducing the BIG-IP Controller system utilities 6-1
sod 6-2
bigd 6-4
big3d 6-9

Chapter 7
BIG/db Configuration Keys

Supported BIG/db configuration keys 7-1
Using the Failover and Cluster keys 7-1

Index

F5 Networks

BIG·IP®

Local Traffic Controller

Introduction

Getting started

Before you start installing the controller, we recommend that you browse the Administrator Guide and find the load balancing solution that most closely addresses your needs. Briefly review the basic configuration tasks and the few pieces of information you should gather in preparation for completing the tasks, such as IP addresses and host names.

Once you find your solution and gather the necessary network information, turn to this Installation Guide for hardware installation instructions, and then return to the Administrator Guide to follow the steps for setting up your chosen solution.

You can use this Reference guide to find basic information about BIG-IP objects, command line syntax, and system configuration files and variable settings.

Choosing a configuration tool

The BIG-IP platform offers both web-based and command line configuration tools, so that users can work in the environment that they are most comfortable with.

The First-Time Boot utility

All users will use the First-Time Boot utility, a wizard that walks you through the initial system set up. The First-Time Boot utility automatically starts the first time you turn the controller on, and it prompts you to enter basic system information including a root password and the IP addresses that will be assigned to the network interfaces. The Installation Guide provides detailed information about the specific pieces of information that the First-Time Boot utility prompts you to enter.

The Configuration utility

The Configuration utility is a web-based administrative application that you use to configure and monitor the load balancing setup on the BIG-IP Controller. Once you complete the installation

instructions described in this guide, you can use the Configuration utility to the configuration steps outlined in the Administrator Guide. In the Configuration utility, you can also monitor current system performance, and download administrative tools such as the SNMP MIB or the SSH client. The Configuration utility requires Netscape Navigator version 4.7 or later, or Microsoft Internet Explorer version 4.1 or later.

The bigpipe and bigtop command line utilities

The bigpipe™ utility is the command line counter-part to the Configuration utility. Using **bigpipe** commands, you can configure virtual servers, open ports to network traffic, and configure a wide variety of features. To monitor the BIG-IP Controller, you can use certain **bigpipe** commands, or you can use the bigtop™ utility, which provides real-time system monitoring. You can use the command line utilities directly on the BIG-IP Controller, or you can execute commands via a remote shell, such as the SSH client (included with the global release only), or a Telnet client (for countries restricted by cryptography export laws). The BIG-IP Controller Reference Guide provides detailed information about command line syntax.

Using the Administrator Kit

The *BIG-IP® Controller Administrator Kit* provides simple steps for quick, basic configuration, and also provides detailed information about more advanced features and tools, such as the **bigpipe** command line utility. The information is organized into the guides described below.

❖ ***Installation Guide***

The Installation Guide walks you through the basic steps needed to get the hardware plugged in and the system connected to the network. Most users turn to this guide only the first time that they set up a BIG-IP Controller. The Installation Guide also covers general network administration issues, such as setting up common network administration tools including Sendmail.

- ***Administrator Guide***
  The Administrator Guide provides examples of common load balancing solutions supported by the particular type of BIG-IP Controller you purchased. For example, in the BIG-IP HA Controller Administrator Guide, you can find everything from a basic web server load balancing solution to a firewall load balancing solution.
- ***Reference Guide***
  The Reference Guide provides basic descriptions of individual BIG-IP objects, such as pools, nodes, and virtual servers. It also provides syntax information for **bigpipe** commands, configuration utilities, configuration files, and system utilities.
- ***F-Secure SSH User Guide***
  This guide is distributed only with BIG-IP Controllers that support the F-Secure SSH client (a tool used for remote command line access). It provides information about setting up and using the SSH client.

Stylistic conventions

To help you easily identify and understand certain types of information, all F5 Networks administrative documentation uses the stylistic conventions described below.

WARNING

All examples in F5 Networks documentation use only non-routable IP addresses. When you set up the solutions we describe, you must use IP addresses suitable to your own network in place of our sample addresses.

Identifying new terms

When we first define a new term, the term is shown in bold italic text. For example, a ***virtual server*** is a the combination of an IP address and port that maps to a set of back-end servers.

Identifying references to objects, names, and commands

We apply bold text to a variety of items to help you easily pick them out of a block of text. These items include web addresses, IP addresses, utility names, and portions of commands, such as variables and keywords. For example, the **bigpipe vip** command requires that you include at least one **<node>** variable.

Identifying references to other documents

We use italic text to denote a reference to another document. In references where we provide the name of a book as well as a specific chapter or section in the book, we show the book name in bold, italic text, and the chapter/section name in italic text to help quickly differentiate the two. For example, you can find information about bigpipe commands in the *bigpipe Command Reference* section of the ***BIG-IP Controller Reference Guide***.

Identifying command syntax

We show actual, complete commands in bold Courier text. Note that we do not include the corresponding screen prompt, unless the command is shown in a figure that depicts an entire command line screen. For example, the following command sets the BIG-IP Controller load balancing mode to Round Robin:

```
bigpipe lb rr
```

Table 1 explains additional special conventions used in command line syntax.

Item in text	Description
****	Continue to the next line without typing a line break.
< >	You enter text for the enclosed item. For example, if the command has **<your name>**, type in your name.
|	Separates parts of a command.
[ ]	Syntax inside the square brackets is optional.
...	Indicates that you can type a series of items.

Table 1 *Command line syntax conventions*

Finding additional help and technical support resources

In addition to this administrator guide, you can find technical documentation about the BIG-IP Controller in the following locations:

- **Release notes**
  The release note for the current version of the BIG-IP Controller is available from the web server on the BIG-IP Controller. The release note contains the latest information for the current version, including a list of new features and enhancements, a list of fixes, and, in some cases, a list of known issues.
- **Online help for BIG-IP Controller features**
  You can find help online in three different locations:
  - The web server on the BIG-IP Controller has PDF versions of the guides included in the Administrator Kit. BIG-IP Controller upgrades replace these guides with updated versions as appropriate.
  - The web-based Configuration utility has online help for each screen. Simply click the **Help** button in the toolbar.
  - Individual **bigpipe** commands have online help, including command syntax and examples, in standard UNIX man page format. Simply type the command followed by the question mark option (**-?**), and the BIG-IP Controller displays the syntax and usage associated with the command.
- **Third-party documentation for software add-ons**
  The web server on the BIG-IP Controller contains online documentation for all third-party software included with the BIG-IP Controller, such as GateD.
- **Technical support via the World Wide Web**
  The F5 Networks Technical Support web site, **http://tech.F5.com**, provides the latest technical notes, answers to frequently asked questions, updates for administrator guides (in PDF format), and the Ask F5 natural language question and answer engine. To access this site, you need to obtain a customer ID and a password from the F5 Help Desk.

What's new in version 3.3

The BIG-IP Controller offers the following major new features in version 3.3, in addition to many smaller enhancements.

BIG-IP e-Commerce Controller

The BIG-IP e-Commerce Controller is a new member of the BIG-IP product family. You can use the BIG-IP e-Commerce Controller to process SSL connections to your network. This controller contains a specific set of software and hardware features that accelerate SSL connections.

BIG-IP Cache Load Balancer

This version of the BIG-IP Controller is available as the BIG-IP Cache Controller. The BIG-IP Cache Controller version contains a specific set of features from the BIG-IP Controller that maximizes the efficiency of caches in your network. In addition to the load balancing features available with this controller, this version of the controller has new rule syntax that provides the ability to redirect HTTP requests to caches in your network. These features include:

- **Cacheable content determination**
  This feature enables you to determine the type of content you cache on the basis of any combination of elements in the header of an HTTP request.
- **Content affinity**
  This feature assures that the same cache serves the same content subset even when caches become temporarily unavailable or when caches are added to or deleted from the cache pool.
- **Hot content load balancing**
  When configured, this feature identifies highly requested content and redirects these requests to a hot pool for load balancing.
- **Intelligent cache population**
  When configured, this feature allows caches to retrieve content from other caches in addition to the origin web server.

Performance enhancements

This version of the BIG-IP Controller includes internal performance enhancements. These enhancements improve the overall performance of the BIG-IP Controller.

Learning more about the BIG-IP Controller product family

The BIG-IP Controller platform offers many different software systems. These systems can be stand-alone, or can run in redundant pairs, with the exception of the BIG-IP e-Commerce Controller, which is only available as a stand-alone system. You can easily upgrade from any special-purpose BIG-IP Controller to the BIG-IP HA Controller, which supports all BIG-IP Controller features.

- **The BIG-IP LB Controller**
  The BIG-IP LB Controller provides basic load balancing features.
- **The BIG-IP FireGuard Controller**
  The BIG-IP FireGuard Controller provides load balancing features that maximize the efficiency and performance of a group of firewalls.
- **The BIG-IP Cache Controller**
  The BIG-IP Cache Controller uses content-aware traffic direction to maximize the efficiency and performance of an group of cache servers.
- **The BIG-IP e-Commerce Controller**
  The BIG-IP e-Commerce Controller uses SSL acceleration technology to increase the speed and reliability of the secure connections that drive e-commerce sites.

❖ **The BIG-IP HA Controller**
The BIG-IP HA Controller provides all features from the basic BIG-IP LB Controller to the advanced BIG-IP FireGuard, BIG-IP Cache Controller, and BIG-IP e-Commerce Controller products.

◆ **Note**

BIG-IP Controllers distributed outside of the United States to a select few countries, regardless of system type, do not support encrypted communications. They do not include the F-Secure SSH client, nor do they support SSL connections to the BIG-IP web server. Instead, you can use the standard Telnet, FTP, and HTTP protocols to connect to the unit and perform administrative functions.

F5 Networks BIG-IP Local Traffic Controller

I

Object Properties

Object properties

This chapter lists the properties of the major objects you can configure on the BIG-IP Controller. The objects described in this chapter include:

- extended application verification (EAV)
- extended content verification (ECV)
- filters
- IP forwarding
- interface
- load balancing
- network address translation (NAT)
- node
- pool
- port
- redundant system
- rule
- secure network address transation (SNAT)
- timer settings
- virtual server

Extended Application Verification (EAV)

Extended Application Verification (EAV) is a sophisticated type of service check typically used to confirm whether an application running on a node is responsive to client requests. To determine whether a node application is responsive, the BIG-IP Controller uses a custom program referred to as an ***external service checker***. An external service checker program essentially provides the option to customize service check functionality for the BIG-IP Controller. It is external to the BIG-IP system itself, and is usually developed by the customer. However, the BIG-IP Controller ships with several external service check programs. These include service check programs for FTP, POP3, SMTP, NNTP, and SQL.

The attributes you can configure for a EAV are in Table 1.7.

Attributes	Description
Custom EAV	You can set up custom EAV service checks on the BIG-IP Controller.
Bundled EAVs	The BIG-IP Controller software includes bundled EAV service check scripts for checking FTP, POP3, SMTP, NNTP, and SQL.

Table 1.1 *The attributes you can configure for EAV.*

You can use an external service checker to verify Internet or intranet applications, such as a web application that retrieves data from a back-end database and displays the data in an HTML page.

An external service checker program works in conjunction with the **bigd** daemon, which verifies node status using node pings and service checks. If you configure external service check on a specific node, the **bigd** daemon checks the node by executing the external service checker program. Once the external service checker executes, the **bigd** daemon looks for output written by the external service checker. If the **bigd** daemon finds output from the external service checker, it marks the node **up**. If it does not find output from the external service checker, it marks the node **down**.

Note that **bigd** does not actually interpret output from the external service checker; it simply verifies that the external service checker created output.

Note

Custom external service checker programs are custom programs that are developed either by the customer, or by the customer in conjunction with F5 Networks.

WARNING

Active checks that look for a receive string only accept 5000 bytes from the server before assuming that the receive string is not in the content.

Setting up custom EAV service checks

An ***Extended Application Verification service check*** is a service check that is performed on an application running on a host on the network connected to the BIG-IP Controller. You can create a custom application for this purpose. Complete the following four tasks to implement a custom EAV service check program on the BIG-IP Controller:

- If you use a custom EAV service check program, verify that your external service checker program meets certain requirements, such as creating a **pid** file.
- Install the external service checker program on the BIG-IP Controller.
- Allow EAV service checks in the BIG-IP configuration.
- Configure the specific nodes to use the EAV service check.

Verifying external service checker requirements

Extended Application Verification (EAV) is intended to provide maximum flexibility. The external service checker programs that you create can use any number of methods to determine whether or

not a service or an application on a node is responsive. The external service checker must, however, meet the following minimum requirements:

- The external service checker must use a **pid** file to hold its process ID, and the **pid** file must use the following naming scheme:

/var/run/pinger.<ip>..<port>.pid.

- As soon as the external service checker starts, if the **pid** file already exists, the external service checker should read the file and send a **SIGKILL** command to the indicated process.
- The external service checker must write its process ID to the **pid** file.
- If the external service checker verifies that the service is available, it must write standard output. If the external service checker verifies that the service is not available, it cannot write standard output.
- The external service checker must delete its **pid** file before it exits.

The BIG-IP Controller includes a several sample external service checker scripts for HTTP, NNTP, SMTP, and POP3. These scripts can be found in the following location:

/usr/local/lib/pingers/sample_pinger

The sample service checker, shown in Figure 1.1, is included with the BIG-IP Controller.

```
# these arguments supplied automatically for all external pingers:
# $1 = IP (nnn.nnn.nnn.nnn notation or hostname)
# $2 = port (decimal, host byte order)
# $3 and higher = additional arguments
#
# In this sample script, $3 is the regular expression
#

pidfile="/var/run/pinger.$1..$2.pid"

if [ -f $pidfile ]
then
   kill -9 `cat $pidfile` > /dev/null 2>&1
fi

echo "$$" > $pidfile

echo "GET /" | /usr/local/lib/pingers/nc $1 $2 2> /dev/null | \ grep -E -
i $3 > /dev/null

status=$?
if [ $status -eq 0 ]
then
    echo "up"
fi
rm -f $pidfile
```

Figure 1.1 *The HTTP external service checker program*

Installing the external service checker on the BIG-IP Controller

To install an EAV service check script, place it in the **/usr/local/lib/pingers** directory. This is the default location for external service checker applications. You can install external service checker applications to other directory locations if desired.

Allowing EAV service checks

Once you install an external service checker on the BIG-IP Controller, you need to add an entry to the /**etc/bigd.conf** file.

To allow external service checking, you need to add the following entry to the **/etc/bigd.conf** file:

```
external  [<node_ip>:]<port> [<path>]<pinger_name> \
   ["<argument_string>"]
```

The <**path**> variable can be an absolute or a relative path to the external checker application. Absolute paths should begin with a slash ("/"). If no path is specified, the default path is substituted (**usr/local/lib/pingers**). The <**pinger_name**> argument is the name of the pinger script to use for service checking.

The **"<argument_string>"** variable must consist of exactly one string in quotation marks. The string may include any number of arguments, delimited in the usual way by white space, for example:

```
external n1:8000 my_pinger "-a 600 -b"
```

In the above example, the BIG-IP Controller runs the script **/usr/local/lib/pingers/my_pinger** to check port 8000, with additional arguments.

In the following example, the BIG-IP Controller checks port 8000 on **n1**. The BIG-IP Controller runs a separate copy of the external service checker named **my_pinger** for each node:

```
external n1:8000 my_pinger "-a -b"
external 8000 my_pinger "-b"
```

In this example, the first entry specifies how to ping port 8000 on node **n1**. The second entry specifies how to ping port 8000 on any other node.

Command line arguments for EAV service checks

The BIG-IP Controller performs the external service check at specified intervals. The BIG-IP Controller actually uses the service ping interval, which is set using the **bigpipe tping_svc** command.

The external service checker runs as root. The BIG-IP Controller starts an external service checker using the following shell command:

```
<path> <pinger name> <node_ip> <port> [ <additional_argument> ... ]
```

For the case of the example shown above, the appropriate command would be:

```
/usr/local/lib/pingers/my_pinger n1 8000 -a 600 -b
```

The BIG-IP Controller inserts the node IP and port number before the additional arguments that are specified in the **/etc/bigd.conf** file.

Note that the standard input and output of an external service checker are connected to **bigd**. The **bigd** does not write anything to the external service checker's standard input, but it does read the external service checker's standard output. If **bigd** is able to read any data from the external service checker program, the particular service is considered **up**.

Using the EAV pingers bundled with the BIG-IP Controller

The BIG-IP Controller includes several sample external service checker scripts for HTTP, NNTP, SMTP, POP3, and SQL. These scripts can be found in this location:

```
/usr/local/lib/pingers/
```

The following sections describe how to set up each of these service checkers.

EAV service check for FTP

This section describes how to set up the BIG-IP Controller to perform EAV service checks on FTP services.

The FTP pinger requires three arguments: a full path to the file on any given server, a user name, and a password. Here are example **bigd.conf** entries:

```
external 10.0.0.57:21 /usr/local/lib/pingers/FTP_pinger
   "/pub/demo/file.txt anonymous user@company.com"
```

```
external 10.0.0.62:21 /usr/local/lib/pingers/FTP_pinger
   "/pub/spool/incoming.doc carol carols_password"
```

The FTP pinger attempts to download the specified file to the **/var/tmp** directory. A successful retrieval of any file with the name indicated is considered a successful ping.

To configure the FTP EAV check in the Configuration utility

1. In the navigation pane, click **Nodes**.
   The Node Properties screen opens.
2. In the **Extended** section, click the EAV box to enable EAV service checking. The service check frequency and service check timeout must be set in order to access this option.
3. In the **Type** list, select the FTP service checker. The External Program Path is automatically filled in when you select a pinger from the list.
4. In the **External Program Arguments** box, type in the arguments required for the FTP service checker: a full path to the file on any given server, a user name, and a password. For example:

   ```
   /pub/demo/file.txt anonymous
   user@company.com
   ```

   ```
   /pub/spool/incoming.doc carol
   carols_password
   ```

5. Click the **Apply** button.

EAV service check for POP3

This section describes how to set up the BIG-IP Controller to perform EAV service checks on POP3 services.

The POP3_pinger for Post Office Protocol requires only two arguments, a user name and a password. This check is considered successful if it successfully connects to the server, logs in as the indicated user, and logs out again. Here are example **bigd.conf** entries:

```
external 10.0.0.57:109 /usr/local/lib/pingers/POP3_pinger "alice
   alices_password"
```

```
external 10.0.0.57:109 /usr/local/lib/pingers/POP3_pinger "bob
   bobs_password"
```

To configure the POP3 EAV check in the Configuration utility

1. In the navigation pane, click **Nodes**.
   The Node Properties screen opens.
2. In the **Extended** section, click the EAV box to enable EAV service checking. The service check frequency and service check timeout must be set in order to access this option.
3. In the **Type** list, select the POP3 service checker. The External Program Path is automatically filled in when you select a service checker from the list.
4. In the **External Program Arguments** box, type in a user name and password. For example:

   `alice alices_password`

   `bob bobs_password`
5. Click the **Apply** button.

EAV service check for SMTP

This section describes how to set up the BIG-IP Controller to perform EAV service checks on SMTP services.

The SMTP_pinger for mail transport servers requires only one argument, a string identifying the server from which the EAV is originating. This is an extremely simple pinger that checks only that the server is up and responding to commands. It counts a success if the mail server it is connecting to responds to the standard SMTP **HELO** and **QUIT** commands. Here is an example **bigd.conf** entry:

```
external 10.0.0.57:25 /usr/local/lib/pingers/SMTP_pinger
   "bigip@internal.net"
```

To configure the SMTP EAV check in the Configuration utility

1. In the navigation pane, click **Nodes**.
   The Node Properties screen opens.

2. In the **Extended** section, click the EAV box to enable EAV service checking. The service check frequency and service check timeout must be set in order to access this option.
3. In the **Type** list, select the SMTP service checker. The External Program Path is automatically filled in when you select a service checker from the list.
4. In the **External Program Arguments** box, type in a string identifying the server from which the EAV is originating. For example:

   `bigip@internal.net`
5. Click the **Apply** button.

EAV service check for NNTP

This section describes how to set up the BIG-IP Controller to perform EAV service checks on NNTP services.

The NNTP_pinger for Usenet News requires only one argument, a newsgroup name to check for presence. If the NNTP server being queried requires authentication, the user name and password can be provided as additional arguments. This pinger counts a success if it successfully retrieves a newsgroup identification line from the server. Here are example **bigd.conf** entries, the second showing the optional login parameters:

```
external 10.0.0.57:119 /usr/local/lib/pingers/NNTP_pinger
   "comp.lang.java"
external 10.0.0.62:119 /usr/local/lib/pingers/NNTP_pinger
   "local.chat username password"
```

To configure the NNTP EAV check in the Configuration utility

1. In the navigation pane, click **Nodes**.
   The Node Properties screen opens.
2. In the **Extended** section, click the EAV box to enable EAV service checking. The service check frequency and service check timeout must be set in order to access this option.

3. In the **Type** list, select the NNTP service checker. The External Program Path is automatically filled in when you select a service checker from the list.
4. In the **External Program Arguments** box, type in the news group name for which you want to check. If the NNTP server being queried requires authentication, you can provide the user name and password as additional arguments. For example:

   ```
   comp.lang.java
   ```

   ```
   local.chat username password
   ```

5. Click the **Apply** button.

EAV service check for SQL-based services

This section describes how to set up the BIG-IP Controller to perform EAV service checks on SQL-based services such as Microsoft SQL Server versions 6.5 and 7.0, and also Sybase.

The service checking is accomplished by performing an SQL login to the service. If the login succeeds, the service is considered **up**, and if it fails, the service is considered **down**. An executable program, **tdslogin** performs the actual login.

1. Test the login manually:

   ```
   cd /usr/local/lib/pingers
   ```

   ```
   ./tdslogin 192.168.1.1 1433 mydata user1
   mypass1
   ```

   Replace the IP address, port, database, user, and password in this example with your own information.

   You should receive the message:

   ```
   Login succeeded!
   ```

   If you receive the connection refused message, verify that the IP and port are correct. See the Troubleshooting SQL based EAV service checks section for more tips.

2. Create an entry in the **/etc/bigd.conf** with the following syntax:

```
external 192.168.1.1:1433
"/usr/local/lib/pingers/SQL_pinger" "mydata
user1 mypass1"
```

In this entry, **mydata** is the name of the database, **user1** is the login name, and **mypass1** is the password.

3. Add entries in the **/etc/bigip.conf** for the service checking:

```
tping_svc 1433 5
```

```
timeout_svc 1433 15
```

4. Reload the **/etc/bigip.conf** and restart **bigd**:

```
bigpipe -f /etc/bigip.conf
```

```
bigd
```

5. Verify that the service check is being performed correctly: If the service is **up**, change the password in **/etc/bigd.conf** to an invalid password and restart **bigd**. The service should go down after the timeout period elapses.

   Correct the password and restart **bigd** and the service should go up again.

Troubleshooting SQL-based service checks

If you are having trouble, you should verify that you can login using another tool. For example, if you have Microsoft NT SQL Server version 6.5, there is a client program **ISQL/w** included with the SQL software. This client program performs simple logins to SQL servers. Use this program to test whether you can login using the ISQL/w program before attempting logins from the BIG-IP Controller.

Creating a test account for Microsoft SQL Server

On the SQL Server, you can run the SQL Enterprise Manager to add logins. When first entering the SQL Enterprise Manager, you may be prompted for the SQL server to manage.

You can register servers by entering the machine name, user name, and password. If these names are correct, the server will be registered and you will be able to click on an icon for the server. When you expand the subtree for the server, there will be an icon for Logins.

Underneath this subtree, you can find the SQL logins. Here, you can change passwords or add new logins by right-clicking on the Logins icon. Click this icon to open an option to **Add login**. After you open this option, enter the user name and password for the new login, as well as which databases the login is allowed to access. You must grant the test account access to the database you specify in the EAV configuration.

Extended Content Verification (ECV)

Extended Content Verification service checking is another feature you can configure after you have performed the three basic configuration tasks. ECV service check is a special type of service checking that actually retrieves content from a server. If the content matches the expected result, the BIG-IP Controller marks the node **up** and uses it for load balancing. If the content does not match, or if the server does not return content, the BIG-IP Controller marks the node **down**, and does not use it for load balancing.

The attributes you can configure for ECV are in Table 1.7

Attributes	Description
Normal ECV	A normal ECV checks for content on the specified server. If the expected content is retrieved, the BIG-IP controller makes the server available for load balancing. If the content is not retrieved, the server is marked down.
SSL ECV	The SSL ECV performs the same service check as a normal ECV, however, it is designed to work with servers that use SSL.
Reverse ECV	A reverse ECV check marks a node unavailable for load balancing when it retrieves the expected content. For example, if the content on your web site is dynamic, you can set up a reverse ECV to check for the string **Error**. A match for this string indicates that the server is down.
Transparent ECV	You can use a transparent ECV to check a transparent node. With transparent ECV, you configure the service check to check content on a node through a transparent device.
Manually creating and testing ECVs	You can manually configure and test ECVs by editing the **/etc/bigd.conf** with a text editor.

Table 1.2 *The attributes you can configure for ECV.*

You can set up ECV service check in the Configuration utility, or you can use a text editor, such as **vi** or **pico**, to manually create the **/etc/bigd.conf** file, which stores ECV information.

ECV service check is most frequently used to verify content on web servers, although you can use it for more advanced applications, such as verifying firewalls or mail servers. This section focuses on setting up ECV for web servers. For details about using advanced ECV service check options, see the ***BIG-IP Controller Administrator Guide***, *Working with Advanced Service Check Options*.

◆ Note

It is important to note that the intervals and timeouts for service checks apply to EAV and ECV service checks. These timeouts are configured by setting the service check timers. For more information about setting these timers, see timer settings, on page 1-138.

ECV service check properties

ECV service check is a property of both a node port and a node. If you define ECV service check settings for a node port, all nodes that use the port inherit the ECV service check settings. You can override these settings by defining ECV service check settings for the node itself.

There are actually three different types of ECV service check settings that you can define:

- **ECV normal**
  An ***ECV normal*** service check requires that the BIG-IP Controller mark a node **up** (available for load balancing) when the retrieved content matches the expected result. For example, if the home page for your web site included the words **Welcome home**, you could set up an ECV service check to look for the string **"Welcome home"**. A match for this string would mean that the web server is up and available.
- **ECV SSL**
  An ***ECV SSL*** service check performs the same function as an ECV normal service check, but it is designed to work with secure servers that use the SSL protocol, rather than standard servers using HTTP. The BIG-IP Controller uses SSL version 3, as do popular web browsers, but it is backward-compatible for web servers that support only version 2.
- **ECV reverse**
  In contrast, an ***ECV reverse*** service check requires that the BIG-IP Controller mark a node **down** (not available for load balancing) when the retrieved content matches the expected result. For example, if the content on your web site home page is

dynamic and changes frequently, you may prefer to set up a reverse ECV service check that looks for the string **"Error"**. A match for this string would mean that the web server was down.

WARNING

When the BIG-IP Controller checks content looking for a match, it reads through the content until the service check times out, or until the read reaches 5,000 bytes, whichever comes first. When you choose text, an HTML tag, or an image name to search for, be sure to pick one that appears in the first 5,000 bytes of the web page.

Writing regular expressions for ECV service checks

When you set up an ECV service check for a web server, you need to define a send string and a receive expression. A ***send string*** is the request that the BIG-IP Controller sends to the web server. Send strings typically request that the server return a specific web page, such as the default page for a web site. For example, the most common send string is **"GET /"** which simply retrieves the default HTML page for a web site. The ***receive expression*** is the text string that the BIG-IP Controller looks for in the returned web page.

Receive expressions use regular expression syntax, and they are not case-sensitive. Although regular expressions can be complex, you will find that simple regular expressions are adequate for most ECV service checks.

The corresponding receive string could be any simple text string included in your home page, such as text, HTML tags, or image names.

Sample send strings

The send string below is probably the most common send string, and it retrieves the default HTML page for a web site. Note that all send strings are enclosed by quotation marks (" ") inside the **/etc/bigd.conf** file.

```
"GET /"
```

To retrieve a specific page from a web site, simply enter a fully qualified path name:

```
"GET /www/support/customer_info_form.html"
```

Sample receive expressions

The most common receive expressions contain a text string that would be included in a particular HTML page on your site. The text string can be regular text, HTML tags, or image names. Note that all receive expressions are enclosed by quotation marks (" ").

For example, the following receive expression attempts to match the text **Welcome**, and it is useful for ECV reverse service checks:

```
"welcome"
```

The sample receive expression below searches for a standard HTML tag. Note that even though you are searching for an HTML tag, you still need to enclose the regular expression with quotation marks (" ").

```
"<HEAD>"
```

You can also use null receive expressions, formatted as the one shown below. When you use a null receive expression, the BIG-IP Controller considers any content retrieved to be a match.

```
""
```

Null receive expressions are suitable only for ECV normal and ECV SSL. Note, however, that if you use them you run the risk of the BIG-IP Controller considering an HTML error page to be a successful service check.

◆ Note

The regular expression syntax discussed here is not the same as the ***wildcard syntax*** *that is commonly used in command shells. For more information about regular expression, see the man page for* ***re_format****. To view the man page for* ***re_format****, type* ***man re_format*** *at the command line.*

Setting up ECV service checks in the Configuration utility

In the Configuration utility, you can set ECV service check options in the Global Node Port Properties screen, and also in individual Node Properties screens. Regardless of which screen you use to configure the options, the steps are the same.

To set up ECV service check in the Configuration utility

1. In the navigation pane, click **Nodes**.
   The Nodes screen opens.
2. Select a node from the list.
   The Node Properties screen opens.
3. If you want to configure ECV service check options, stay in this screen. If you want to configure ECV service check options for the port that the node uses, click the port number listed next to the IP address of the node.
4. Click the **ECV** button.
5. In the **Type** box, choose the type of ECV service check you want to set up: normal, reverse, or SSL.
6. In the **Send String/Destination String** box, type the send string that requests the web page. Note that the Configuration utility automatically places quotation marks around the string itself. For example, the following string retrieves the default HTML page for the site.

   ```
   GET /
   ```

7. In the **Receive Rule** box, type the receive expression that the BIG-IP Controller should look for in the returned web page. For example, the following receive expression looks for a text string in a web page:

   ```
   Welcome home!
   ```

8. Click the **Apply** button.

Setting up ECV service checks for transparent nodes

In addition to verifying content on web servers, you can use Extended Content Verification (ECV) service checks to verify connections to mail servers and FTP servers through transparent nodes. If you want to set up ECV service checks through a transparent node to these types of servers, there are certain special issues that you need to address.

Configuring ECV for transparent nodes

You can set up ECV to verify that a transparent node is functioning properly. To check if a transparent node is functioning, you can add an entry to the **/etc/bigd.conf** file that allows you to retrieve content through the transparent node.

You can use a text editor, such as **vi** or **pico**, to manually create the **/etc/bigd.conf** file, which stores ECV information. To create the entry for checking a transparent node, use the following syntax:

```
transparent <node ip>:<node port> <url> ["recv_expr"]
```

You can also use the following syntax for this entry:

```
transparent <node ip>:<node port> <dest ip>:<dest port>/<path>
   ["recv_expr"]
```

For example, if you want to run a service check through the transparent firewall 10.10.10.101:80 to the node 10.10.10.53:80, the entry might look like this:

```
transparent 10.10.10.101:80 10.10.10.53:80/www/forms/survey.html
   "Company Survey"
```

For more information about these configuration entries, please refer to Table 1.3.

Configuration Entry	Description
`transparent`	The **transparent** keyword is required at the beginning of the entry.
`node ip`	The IP address, in dotted decimal notation, of the transparent firewall or proxy. This IP cannot be a wild card IP (0.0.0.0). Note that the node must be defined as a node in a pool definition. Typically this would be a wild card virtual server (0.0.0.0). This entry can also be specified as a fully qualified domain name (FQDN). In order to use an FQDN, the BIG-IP Controller must be configured for name resolution.
`node port`	This entry is the node port to use for the ECV check. This port can be zero. This entry can be numeric or can use a well-known service name, such as **http**.
`dest ip:dest port`	This is the combination of the destination, in dotted decimal notation, and port number of the destination against which the ECV service check is performed. The IP address cannot be a wild card (0.0.0.0). The port number is optional. The port can be specified as any non-zero numeric port number, or specified as a well-known port name, such as **http**.
`url`	The URL is an optional standard HTTP URL. If you do not specify a URL, a default URL is retrieved using the HTTP 1.0 request format. This entry can also be specified using a complete URL with an embedded FQDN. This entry cannot be longer than 4096 bytes. In order to resolve an FQDN, the BIG-IP Controller must be configured for name resolution.
`recv string`	This string is optional. If you specify a string, the string you specify is used to perform standard ECV verification. This entry must be enclosed in quotation marks, and cannot be longer than 128 bytes.

Table 1.3 *ECV configuration entries.*

◆ Note

*The **/etc/bigd.conf** file is read once at startup. If you change the file on the command line, you must reboot or restart **bigd** for the changes to take effect. To restart **bigd** from the command line, type **bigd**. If you make changes in the Configuration utility, clicking the **Apply** button makes changes and restarts **bigd**. For more information, see bigd, on page 6-4.*

Setting up ECV through transparent nodes with the Configuration utility

New ECV syntax facilitates using ECV with transparent nodes. With it, you can test whether a transparent node is functioning properly by retrieving content through it. You can enable this feature in the Configuration utility or from the command line. This section describes how to enable this feature from the Configuration utility.

◆ Note

You must have at least one wildcard virtual server configured in order to configure ECV through a transparent node.

To set up ECV through a transparent node using the Configuration utility

There are two procedures required to set up ECV through a transparent node. First, set up the frequency and timeout for the port:

1. In the navigation pane, click expand button (+) next to **Nodes**.
   The navigation tree expands to display **Ports**.
2. In the navigation pane, click **Ports**.
   The Global Node Port properties screen opens.
3. In the **Port** list, click the port you want to configure.
   The properties screen for the port opens.
4. In the **Frequency (seconds)** box, type in the interval (in seconds) at which the BIG-IP Controller performs a service check on the node.
5. In the **Timeout (seconds)** box, type in the time limit (in seconds) that a node has to respond to a service check issued by the BIG-IP Controller.
6. Click the **Apply** button.

After you configure the frequency and timeout settings for the port, set the specific settings for the transparent node:

1. In the navigation pane, click **Nodes**.
   The Node Properties screen opens.
2. In the **Node** list, click the node you want to configure.
   The Node Properties screen opens.
3. In the **Service Check Extended** section, click the **ECV** button to enable ECV.
4. In the **Type** list, select **Transparent**.
   By default, the list is set to Transparent.
5. In the **Send String/Destination String** box, you must type the destination IP address of the node you are checking on the other side of the transparent device. The port number/port name argument is optional. The URL entry is also optional. For more information about what to type in this box, see Table 1.3.
6. In the **Receive Rule** box, you can type an ECV check receive string. The receive string is optional.
7. Click the **Apply** button.

Manually configuring and testing the /etc/bigd.conf file

You can set up ECV service check on the command line by creating an **/etc/bigd.conf** file in a text editor such as **vi** or **pico**. Each line in the **/etc/bigd.conf** file defines a send string and a receive expression for one node, or for one port. Remember that when you define a ECV service check for a port, all nodes that use the port inherit the service check settings.

Changes to the **/etc/bigd.conf** do not take effect until the system is rebooted, or **bigd** is restarted. To restart **bigd**, simply run the command **bigd**.

Setting up the /etc/bigd.conf file

The **/etc/bigd.conf** file uses three different types of syntax for lines in the file that correspond to the three different types of service check that you can configure: ECV normal, ECV SSL, and ECV reverse. The following sections describe the syntax for each type, and provide some useful examples.

To set up an ECV normal service check

The line for a normal ECV service check begins with the keyword **active**. The <**node IP**> parameter is optional, and you need to include it only if you are defining ECV service check for a specific node.

```
active [<node IP>:]<port> "<send_string>" "<recv_expr>"
```

For example, the following line sets up a normal ECV service check for a node, where the BIG-IP Controller looks for the text **Welcome** in the default page for the site.

```
active 192.168.100.10:80 "GET /" "welcome"
```

To set up an ECV SSL service check

The line for an SSL ECV service check begins with the keyword **ssl**. The <**node IP**> parameter is optional, and you need to include it only if you are defining ECV service check for a specific node.

```
ssl [<node IP>:]<port> "<send_string>" "<recv_expr>"
```

For example, the following line sets up an SSL ECV service check for a node port. Note that the receive expression is null. When you use a null receive expression, the BIG-IP Controller considers any retrieved content to be a match.

```
ssl 443 "GET /www/orders/order_form.html" ""
```

To set up ECV reverse service check

The line for a reverse ECV service check begins with the keyword **reverse**. The <**node IP**> parameter is optional, and you need to include it only if you are defining ECV service check for a specific node.

```
reverse [<node IP>:]<port> "<send_string>" "<recv_expr>"
```

For example, the following line sets up a reverse ECV service check for a node port. Note that the receive expression is null. When you use a null receive expression, the BIG-IP Controller considers any retrieved content to be a match.

```
reverse 80 "GET /" ""
```

Testing /etc/bigd.conf syntax

To test /etc/bigd.conf syntax

You can test your ECV syntax in the **bigd.conf** file using the following **bigd** command:

```
/sbin/bigd -d
```

This command parses the file, checks ECV syntax, reports any errors, and then exits.

◆ Note

*The **/etc/bigd.conf** file is read once at startup. If you change the file on the command line, you must reboot or restart **bigd** for the changes to take effect. If you make changes in the Configuration utility, clicking the **Apply** button makes changes and restarts **bigd**. For more information about **bigd**, see the **BIG-IP Controller Reference Guide**, System Utilities.*

Filters

Filters control network traffic by setting whether packets are forwarded or rejected at the external network interface. Filters apply to both incoming and outgoing traffic. When creating a filter, you define criteria which are applied to each packet that is processed by the BIG-IP Controller. You can configure the BIG-IP Controller to forward or block each packet based on whether or not the packet matches the criteria.

The BIG-IP Controller supports two types of filters, IP filters and rate filters.

The attributes you can configure for a filter are in Table 1.4

Filter Attributes	Description
IP filter	You can configure IP filters to control request sent to the BIG-IP Controller by other hosts in the network.
Rate filter	You can configure rate filters to control the flow of traffic into the BIG-IP Controller based on rate classes you define. In order to create a rate filter, you must first define a rate class.
Rate class	You can define a rate class for use with a rate filter. A rate class is a definition used by a rate filter to restrict the flow of traffic into the BIG-IP Controller.

Table 1.4 *The attributes you can configure for a filter.*

IP filters

Typical criteria that you define in IP filters are packet source IP addresses, packet destination IP addresses, and upper-layer protocol of the packet. However, each protocol has its own specific set of criteria that can be defined.

For a single filter, you can define multiple criteria in multiple, separate statements. Each of these statements should reference the same identifying name or number, to tie the statements to the same filter. You can have as many criteria statements as you want,

limited only by the available memory. Of course, the more statements you have, the more difficult it is to understand and maintain your filters.

Configuring IP filters

When you define an IP filter, you can filter traffic in two ways:

- You can filter traffic going to a specific destination or coming from a specific destination, or both.
- The filter can allow network traffic through, or it can reject network traffic.

Defining an IP filter in the Configuration utility

1. In the navigation pane, click **IP Filters**.
   The IP Filters screen opens.
2. In the IP Filters screen, click **Add Filter**.
   The Add IP Filter screen opens.
3. On the Add IP Filter screen, in the **Name** box, type a filter name.
4. From the **Type** list, choose **Accept Packet** to allow traffic, or **Deny Packet** to reject traffic.
5. In the **Source IP Address** box, only if you want the filter to be applied to network traffic based on its source, enter the IP address from which you want to filter traffic.
6. In the **Source Port** box, only if you want the filter to be applied to network traffic based on its source, enter the port number from which you want to filter traffic.
7. In the **Destination IP Address** box, enter the IP address to which you want to filter traffic, only if you want the filter to be applied to network traffic based on its destination.
8. In the **Destination Port** box, enter the port number to which you want to filter traffic, only if you want the filter to be applied to network traffic based on its destination.

9. Click **Add** to add the IP filter to the system.

Note

*For information on configuring IP filters on the command line, refer to the IPFW man page by typing **man ipfw** on the command line. You can configure more complex filtering through the IPFW command line interface.*

Rate filters and rate classes

In addition to IP filters, you can also define rates of access by using a rate filter. Rate filters consist of the basic filter and a rate class. Rate classes define how many bits per second are allowed per connection and the number of packets in a queue.

Configuring rate filters and rate classes

Rate filters are a type of extended IP filter. They use the same IP filter method, but they apply a ***rate class*** which determines the volume of network traffic allowed through the filter.

Tip

You must define at least one rate class in order to apply a rate filter.

Rate filters are useful for sites that have preferred clients. For example, an e-commerce site may want to set a higher throughput for preferred customers, and a lower throughput for random site traffic.

Configuring rate filters involves both creating a rate filter and a rate class. When you configure rate filters, you can use existing rate classes. However, if you want a new rate filter to use a new rate class, you must configure the new rate class before you configure the new rate filter.

To configure a new rate class in the Configuration utility

1. In the navigation pane, click **Rate Filters**.
   The Rate Filters screen opens.
2. In the Rate Filters screen, click **Add Class**.
   The Rate Class screen opens.
3. On the Rate Class screen, in the **Name** box, type a rate class name.
4. In the **Bits Per Second Allowed** box, enter the maximum number of bits per second that you want the class to allow.
5. In the **Minimum Number of Bits Outstanding** box, enter the minimum number of bits required to be sent for processing from the queue at one time.
6. In the **Queue Length (in Packets)** box, enter the maximum number of packets allowed in the queue. Once the BIG-IP Controller fills the queue, it begins to drop subsequent packets received.
7. Click **Add** to add the rate class to the system.

◆ Note

For information on configuring IP filters on the command line, refer to the IPFW man page.

After you have added a rate class, you can configure rate filters for your system.

To configure a rate filter in the Configuration utility

1. Click **Rate Filters** in the navigation pane.
   The Rate Filters screen opens.
2. In the Rate Filters screen, click **Add Class**.
   The Add Class screen opens.
3. On the Rate Filter screen, in the **Name** box, type a name for the rate filter.
4. From the **Rate Class** list, choose a rate class. Note that you must have a rate class defined before you can proceed.

5. In the **Source IP Address** box, enter the IP address from which you want to filter traffic, only if you want the filter to be applied to network traffic based on its source.
6. In the **Source Port** box, enter the port number from which you want to filter traffic, only if you want the filter to be applied to network traffic based on its source.
7. In the **Destination IP Address** box, enter the IP address to which you want to filter traffic, only if you want the filter to be applied to network traffic based on its destination.
8. In the **Destination Port** box, enter the port number to which you want to filter traffic, only if you want the filter to be applied to network traffic based on its destination.
9. Click the **Add** button.

Note

For information on configuring IP filters on the command line, refer to the IPFW man page.

IP forwarding

IP forwarding does not translate node addresses. Instead, it simply exposes the node's IP address to the BIG-IP Controller's external network and clients can use it as a standard routable address. When you turn IP forwarding on, the BIG-IP Controller acts as a router when it receives connection requests for node addresses. IP forwarding does not provide security features, but you can use the IP filter feature to implement a layer of security that can help protect your nodes.

The attributes you can configure for IP forwarding are in Table 1.5

Attributes	Description
Enable IP forwarding globally	You can turn IP forwarding on for the BIG-IP Controller globally either with the Configuration utility, or by turning on the **sysctl** variable **net.inet.ip.forwarding**.
Addressing routing issues	If you turn on IP forwarding, you need to route packets to the node addresses through the BIG-IP Controller.
Enable IP forwarding for a virtual server	Instead of turning IP forwarding on globally, you can create a special virtual server with IP forwarding on.

Table 1.5 *The attributes you can configure for IP forwarding.*

◆ Note

NATs and SNATs do not support the NT Domain or CORBA protocols. Instead of using NATs or SNATs, you need to configure IP forwarding.

Setting up IP forwarding

If you do not want to translate addresses with a NAT or SNAT, you can use the IP forwarding configuration option. IP forwarding is an alternate way of allowing nodes to initiate or receive direct

connections from the BIG-IP Controller's external network. IP forwarding exposes all of the node IP addresses to the external network, making them routable on that network. If your network uses the NT Domain or CORBA protocols, IP forwarding is an option for direct access to nodes.

To set up IP forwarding, you need to complete two tasks:

- **Turn IP forwarding on**
  The BIG-IP Controller uses a system control variable to control IP forwarding, and its default setting is **off**.
- **Verify the routing configuration**
  You probably have to change the routing table for the router on the BIG-IP Controller's external network. The router needs to direct packets for nodes to the BIG-IP Controller, which in turn directs the packets to the nodes themselves.

Turning on IP forwarding

IP forwarding is a property of the BIG-IP Controller system, and it is controlled by the system control variable **net.inet.ip.forwarding**.

To set the IP forwarding system control variable in the Configuration utility

1. In the navigation pane, click the BIG-IP Controller icon.
   The BIG-IP System Properties screen opens.
2. On the toolbar, click **Advanced Properties**.
   The BIG-IP System Control Variables screen opens.
3. Check the **Allow IP Forwarding** box.
4. Click the **Apply** button.

To set the IP forwarding system control variable on the command line

Use the standard **sysctl** command to set the variable. The default setting for the variable is **0**, which is **off**. You want to change the setting to **1**, which is **on**:

```
sysctl -w net.inet.ip.forwarding=1
```

To permanently set this value, you can use a text editor, such as **vi** or **pico**, to manually edit the **/etc/rc.sysctl** file. For additional information about editing this file, see *Setting BIG-IP system control variables*, on page 5-1.

Addressing routing issues for IP forwarding

Once you turn on IP forwarding, you probably need to change the routing table on the default router. Packets for the node addresses need to be routed through the BIG-IP Controller. For details about changing the routing table, refer to your router's documentation.

Configuring forwarding virtual servers

A forwarding virtual server is just like other virtual servers, except that the virtual server has no nodes to load balance. It simply forwards the packet directly to the node. Connections are added, tracked, and reaped just as with other virtual servers. You can also view statistics for forwarding virtual servers.

To configure forwarding virtual servers in the Configuration utility

1. In the navigation pane, click **Virtual Servers**.
   The Virtual Servers screen opens.
2. In the toolbar, click the **Add Virtual Server** button.
   The add virtual server screen opens.
   Type in the virtual server attributes including address and port. Use the IP address/port combination guidelines in the previous section, *To configure a forwarding virtual server from the command line*, to determine what these entries should be.
3. In Resources, click the **Forwarding** button.
4. Click the **Apply** button.

To configure a forwarding virtual server from the command line

Use the following syntax to configure forwarding virtual servers:

bigpipe vip <vip>:<port> [netmask <netmask>] forward

For example, to allow only one service in:

bigpipe vip 206.32.11.6:80 forward

Use the following command to allow only one server in:

bigpipe vip 206.32.11.5:0 forward

To forward all traffic:

bigpipe vip 0.0.0.0:0 forward

Currently, there can be only one wildcard virtual server, whether that is a forwarding virtual server or not. In some of the configurations described here, you need to set up a wildcard virtual server on one side of the BIG-IP Controller to load balance connections across transparent devices. Another wildcard virtual server is required on the other side of the BIG-IP Controller to forward packets to virtual servers receiving connections from the transparent devices and forward them to their destination. You can use another new feature, per-connection routing, with forwarding virtual servers, to route connections back through the device from which the connection originated. In these configurations, you need to create a forwarding virtual server for each possible destination network or host if a wildcard virtual server is already defined to handle traffic coming from the other direction.

Interface

You can use interface attributes to configure how traffic flows through the BIG-IP Controller. Most configuration require you to set the attributes of one or more interface on the BIG-IP Controller.

The attributes you can configure for an interface are in Table 1.6

Interface Attributes	Description
Source processing	You can use this attribute to configure an interface to allow source translation and routing for NATs, virtual server connections, and SNAT connections. This attribute also initiates SNAT connections for packets arriving on this interface. This attribute is a feature of versatile interface configuration.
Destination processing	You can configure this attribute on an interface to allow packets arriving on an interface to be destination translated and routed according current state of a NAT, SNAT, or virtual server connection. This attribute also initiates virtual server connections for packets arriving on this interface. This attribute is a feature of versatile interface configuration.
Source translation	You can configure this attribute on an interface to allow packets arriving on an interface to be source translated and routed according current state of a SNAT or virtual server connection. SNAT destination and source translation occur when a matching SNAT connection exists. This attribute is a feature of versatile interface configuration.
Interface security	You can configure security at the interface level. This attribute is a feature of versatile interface configuration.
Interface failsafe	Use this attribute for redundant fail-over. When you arm interface fail-safe, the controller automatically fails over if it detects there is no traffic on the interface specified.
MAC masquerade	You can use this attribute to set up a media access control (MAC) address that is shared by redundant controllers. This allows you to use the BIG-IP Controllers in a topology with secure hubs.
VLAN tags	You can assign VLAN tags to an interface. You can use VLAN tags to divide a single physical network into additional virtual networks. The VLAN tag determines which virtual network handles the traffic.

Table 1.6 *The attributes you can configure for an interface*

Versatile interface configuration

The versatile interfaces option adds more flexibility for configuring interfaces. You can now change both the source address or destination address and/or route of an IP packet.

In previous versions of the BIG-IP Controller, interfaces were designated as internal or external. With this version of the BIG-IP Controller you can configure specific interface properties based on the properties in Table 1.7.

Interface type	Interface properties
Internal	Processes source addresses Administrative ports open
External	Processes destination addresses Administrative ports locked down

Table 1.7 *The properties for internal and external interfaces*

The ability to change the source or destination can be turned on independently. Essentially, this means you can configure an interface so that it handles traffic going to virtual servers and, independently, you can configure the interface to handle traffic coming in from nodes. You can configure virtual servers and nodes on each interface installed on the BIG-IP Controller. This allows for the most flexible processing of packets by the BIG-IP Controller. When either the source or destination processing feature is turned off on an interface, there is a gain in performance.

When you enable destination processing on a BIG-IP Controller interface, the interface functions in the following manner:

- When the destination address and port on the packet is a virtual server connection, then the interface routes the packet to the Node that is handling the connection that the packet is a part of, picking one if necessary, and depending on how the virtual server is configured, the interface translates the destination address to the node address.

- When the destination address on the packet is the external, or translated, address of a NAT, then the interface translates the destination address to the internal address of the NAT.
- When the destination address on the packet is the external, or translated, address and port of a SNAT connection, then the interface translates the destination address to the original address from which the SNAT connection originated.

When you enable source processing on a BIG-IP Controller interface, the interface functions in the following manner:

- When the source address on the packet is a node, and the packets are destined to a client for whom there is an existing virtual server connection, and depending on how the virtual server is configured, the interface translates the source address to the address and port of the virtual server.
- When the source address on the packet is the original address of a NAT, then the interface translates the source address to the translated address of the NAT.
- When the source address on the packet is the original address and port of a SNAT connection, then the interface translates the source address and port to the translated address of the SNAT.

You can turn on both source and destination processing for an interface. This is possible because their functions do not overlap. For example, a NAT changes the source address on packets coming from clients so that they look like they have a different IP address, and virtual servers change the destination address to load balance the destination. There is no reason why you cannot do both the NAT translation and the virtual server translation. There are some combinations of virtual server and NAT source processing and virtual server and NAT destination processing that do not make sense. For example, if a virtual server processes a packet during source processing, the packet is not handled by virtual server destination processing. Also, if a virtual server processes a packet during destination processing, the packet is not handled by virtual server source processing.

Destination route and translation processing

When destination processing is enabled on an interface, the BIG-IP Controller processes packets arriving at the interface when those packets are addressed to a virtual server, SNAT, or NAT translated address.

It is useful to note that there are two independent activities associated with destination processing: routing and translation. For example, wildcard virtual servers load balance connections across transparent network devices (such as a router or firewall), but they do not perform translation. In fact, translation can be turned off for all virtual servers. Also, with the new forwarding virtual servers, neither next hop load balancing nor translation will occur for connections. These virtual servers only forward packets and so connections can pass through BIG-IP Controller without being manipulated in any way.

When you plan which type of processing to use in the BIG-IP Controller configuration, consider these questions:

- What traffic is translated?
- When those packets reach the BIG-IP Controller interface, does the source IP address, or destination IP address need to be translated?
- Which connections are load balanced across multiple devices?

These questions help identify what kind of processing is required for the network interfaces on the BIG-IP Controller.

Source translation processing

When source translation processing is enabled on an interface, then the BIG-IP Controller processes packets arriving at the interface when those packets are coming from a node, SNAT, or NAT internal address. In this situation, the interface rewrites the source address of the IP packet, changing it from the real server's IP address, or original NAT address, to the virtual server or translated NAT address, respectively. Also, when the last hop feature is

enabled on a virtual server, the packet is routed back to the network device that first transmitted the connection request to the virtual server.

To configure source and destination processing in the Configuration utility

1. In the navigation pane, click **NICs**.
   The Network Interface Cards screen opens. You can view the current settings for each interface in the Network Interface Card table.
2. In the Network Interface Card table, click the name of the interface you want to configure.
   The Network Interface Card Properties screen opens.
   - To enable source processing for this interface, click the **Enable Source Processing** check box.
   - To enable destination processing for this interface, click the **Enable Destination Processing** check box.
3. Click the **Apply** button.

To configure source and destination processing from the command line

Use the following syntax to configure source and destination processing on the specified interface:

```
bigpipe interface <interface> dest [ enable |
    disable ]
bigpipe interface <interface> source [ enable |
    disable ]
bigpipe interface <interface> source_translation [
    enable | disable ]
```

The following example command enables destination processing on the interface **exp0**:

```
bigpipe interface exp0 dest enable
```

The following example command enables source processing on the interface **exp1**:

```
bigpipe interface exp1 source enable
```

Source translation

In a situation where you have an origin cache server on the network external to the BIG-IP Controller, you must configure a default SNAT, enable source translation on the external interface, and set the origin server node address to remote in addition to creating a cache rule. This section describes how to enable source translation on an interface.

To enable source translation on the external interface, type the following command:

```
bigpipe interface <ext_interface> source_translation enable
```

Substitute the name of the external interface for **<ext_interface>**.

Interface security

You can use the **adminport** option to control the security on an interface. The **lockdown** keyword configures the port lockdown used in previous versions of the BIG-IP Controller on the specified interface. If you use this option when you configure an interface, only ports essential to the configuration and operation of BIG-IP Controller and 3DNS Controller are opened. The **open** keyword allows all connections to and from BIG-IP Controller through the interface you specify.

To configure interface security in the Configuration utility

1. In the navigation pane, click **NICs**.
   The Network Interface Cards screen opens. You can view the current settings for each interface in the Network Interface Card table.
2. In the Network Interface Card table, click the name of the interface you want to configure.
   The Network Interface Card Properties screen opens.
3. To set the administration properties, click the Enable Admin list. Choose one of the following options:

- **Lockdown**
  Choose this option to lock down all ports except the ports used for administrative access on this interface.
- **Open**
  Choose this option to open allow connections to all ports on this interface.

4. Click the **Apply** button.

To configure interface security from the command line

Use the following syntax to configure interface security on the specified interface:

```
bigpipe interface <interface> adminport lockdown
bigpipe interface <interface> adminport open
```

Use the following example command to lock down connections to all ports except the administration ports on **exp0**:

```
bigpipe interface exp0 adminport lockdown
```

Use the following example command to allow connections to all ports on **exp1**:

```
bigpipe interface exp1 adminport open
```

WARNING

Use caution when redefining interfaces. When you reconfigure interfaces, make sure that you have set up the interfaces you need for operation. It is possible to accidentally take the controller out of network service by redefining interfaces.

Displaying status for interfaces

Use the following syntax to display the current status and the settings for all installed interface cards:

```
bigpipe interface show
```

Figure 1.2 is an example of the output you see when you issue this command on an active/standby controller in active mode.

```
exp0         11.11.11.2, dest enable, source disable, disarmed, timeout 30
 shared alias 11.11.11.3  netmask 255.0.0.0  broadcast 11.255.255.255  unit 1
exp1         11.12.11.2, dest disable, source enable, disarmed, timeout 30
 shared alias 11.12.11.3  netmask 255.0.0.0  broadcast 11.255.255.255  unit 1
```

Figure 1.2 *The* ***bigpipe interface show*** *command output*

Use the following syntax to display the current status and the setting for a specific interface.

`bigpipe interface <ifname> show`

Arming and disarming the fail-safe mode

Use the following command to activate the BIG-IP Controller interface fail-safe mode.

`bigpipe interface <ifname>  arm`

When armed, the active controller automatically fails over to the standby controller whenever the active controller detects that there is no activity on the specified interface, and subsequently detects no activity on the interface in response to ARP requests. The default fail-safe mode is set to **disarm**.

WARNING

You should arm the fail-safe mode only after you configure the BIG-IP Controller, and both the active and standby units are ready to be placed into a production environment.

Note that you must specify a default route before using the **bigpipe interface failsafe** command. You specify the default route in the **/etc/hosts** and **/etc/netstart** files.

Use the following command to deactivate the BIG-IP Controller interface fail-safe mode.

`bigpipe interface <ifname> failsafe disarm`

Setting the fail-safe timeout

Use the following syntax to set the amount of time, in seconds, that an interface will be monitored for activity in response to a BIG-IP Controller ARP request, in order to be designated operational.

```
bigpipe interface <ifname> timeout <seconds>
```

If no activity is detected on the interface within the specified time, the BIG-IP Controller assumes that the interface is down. Note that the default setting is 30 seconds.

Warning messages and ARP requests are generated after half of the specified time-out period. In the case of an armed BIG-IP Controller in a BIG-IP redundant system, traffic is switched from the active unit to the standby unit at the end of the time-out period. Note that the fail-safe timeout is used only if the fail-safe option is armed on the interface.

Viewing the timeout setting

Use the following syntax to view the fail-over timeout setting for a specific interface:

```
bigpipe interface <ifname> timeout show
```

Setting the MAC masquerade address

Sharing the MAC masquerade address makes it possible to use BIG-IP Controllers in a network topology using secure hubs. You can view the media access control (MAC) address on a given controller using the following command:

```
/sbin/ifconfig -a
```

Use the following syntax to set the MAC masquerade address that will be shared by both BIG-IP Controllers in the redundant system.

```
bigpipe interface <ifname> mac_masq <MAC addr>
```

WARNING

*You must specify a default route before using the **mac_masq** command. You specify the default route in the **/etc/hosts** and **/etc/netstart** files.*

Find the MAC address on both the active and standby units and choose one that is similar but unique. A safe technique for choosing the shared MAC address follows:

Suppose you want to set up **mac_masq** on the external interfaces. Using the **ifconfig -a** command on the active and standby units, you note that their MAC addresses are:

```
Active: exp0 = 0:0:0:ac:4c:a2
Standby: exp0 = 0:0:0:ad:4d:f3
```

In order to avoid packet collisions, you now must choose a unique MAC address. The safest way to do this is to select one of the addresses and logically **OR** the first byte with **0x40**. This makes the MAC address a locally administered MAC address.

In this example, either 40:0:0:ac:4c:a2 or 40:0:0:ad:4d:f3 would be a suitable shared MAC address to use on both BIG-IP Controllers in the redundant system.

The shared MAC address is used only when the BIG-IP Controller is in active mode. When the unit is in standby mode, the original MAC address of the network card is used.

If you do not configure **mac_masq**, on startup, or when transitioning from standby mode to active mode, the BIG-IP Controller sends gratuitous ARP requests to notify the default router and other machines on the local Ethernet segment that its MAC address has changed. See RFC 826 for more details on ARP.

Note

You can use the same technique to configure a shared MAC address for each interface.

Enabling VLAN tags for an interface

To use IEEE 802.1q VLAN Trunk mode, you must first set up VLAN tags in **/etc/netstart** and the shared IP in BIG/db. For detailed information about setting up VLAN tags, see the ***BIG-IP Controller Administrator Guide***, *Using Advanced Network Configurations*.

Use the following syntax to enable, disable, or show the VLAN status of the specified internal interface:

```
bigpipe interface <ifname> vlans enable | disable | show
```

Load Balancing

Load balancing is an integral part of the BIG-IP Controller. A load balancing mode defines, in part, the logic that a BIG-IP Controller uses to determine which node should receive a connection hosted by a particular virtual server.

The load balancing attributes you can configure for the BIG-IP Controller Table 1.8

Load Balancing Attributes	Description
Load balancing modes	You can configure a specific type of load balancing for the BIG-IP Controller.
Changing global load balancing modes	You can set a global load balancing mode on the BIG-IP Controller. The global method is used by all pools that do not have a load balancing method defined.
Using load balancing pools	You must define a specific load balancing method for a pool. You can have various pools configured with different load balancing methods.

Table 1.8 *The load balancing attributes*

The BIG-IP Controller supports specialized load balancing modes that dynamically distribute the connection load, rather than following a static distribution pattern such as Round Robin. Dynamic distribution of the connection load is based on various aspects of real-time server performance analysis, such as the current number of connections per node or the fastest node response time. The following section describes how each load balancing mode distributes connections, as well as how to set the load balancing mode on the BIG-IP Controller. The global load balancing method is not saved as part of BIG-IP Controller configuration. When you define a global method, the global method is set for any pool with an **appgen_** name prefix.

The default global load balancing mode on the BIG-IP Controller is Round Robin, and it simply passes each new connection request to the next server in line, eventually distributing connections evenly

across the array of machines being load balanced. Round Robin mode works well in most configurations, especially if the equipment that you are load balancing is roughly equal in processing speed and memory. If you want to use the Round Robin load balancing mode, you can skip this section, and begin configuring features that you want to add to the basic configuration.

However, if you are working with servers that differ significantly in processing speed and memory, you may want to switch to Ratio load balancing mode. In Ratio mode, the BIG-IP Controller distributes connections among machines according to ratio weights that you define, where the number of connections that each machine receives over time is proportionate to the ratio weight you define for each machine.

◆ Tip

*The default ratio weight for a node is **1**. If you keep the default ratio weight for each node in a virtual server mapping, the nodes receive an equal proportion of connections as though you were using Round Robin load balancing.*

Understanding individual load balancing modes

Individual load balancing modes take into account one or more dynamic factors, such as current connection count. Because each application of the BIG-IP Controller is unique, and node performance depends on a number of different factors, we recommend that you experiment with different load balancing modes, and choose the one that offers the best performance in your particular environment.

Round Robin

Round Robin passes each new connection request to the next server in line, eventually distributing connections evenly across the array of machines being load balanced. Round Robin mode works well in most configurations, especially if the equipment that you are load balancing is roughly equal in processing speed and memory.

Ratio

In Ratio mode, the BIG-IP Controller distributes connections among machines according to ratio weights that you define, where the number of connections that each machine receives over time is proportionate to the ratio weight you define for each machine.

Fastest mode

Fastest mode passes a new connection based on the fastest response of all currently active nodes. Fastest mode may be particularly useful in environments where nodes are distributed across different logical networks.

Least Connections mode

Least Connections mode is relatively simple in that the BIG-IP Controller passes a new connection to the node with the least number of current connections. Least Connections mode works best in environments where the servers or other equipment you are load balancing have similar capabilities.

Observed mode

Observed mode uses a combination of the logic used in the Least Connection and Fastest modes. In Observed mode, nodes are ranked based on a combination of the number of current connections and the response time. Nodes that have a better balance of fewest connections and fastest response time receive the a greater proportion of the connections. Observed mode also works well in any environment, but may be particularly useful in environments where node performance varies significantly.

Predictive mode

Predictive mode also uses the ranking methods used by Observed mode, where nodes are rated according to a combination of the number of current connections and the response time. However, in

Predictive mode, the BIG-IP Controller analyzes the trend of the ranking over time, determining whether a node's performance is currently improving or declining. The nodes with better performance rankings that are currently improving, rather than declining, receive a higher proportion of the connections. Predictive mode works well in any environment.

Priority mode

Priority mode is a special type of round robin load balancing. In Priority mode, you define groups of nodes and assign a priority level to each group. The BIG-IP Controller begins distributing connections in a round robin fashion to all nodes in the highest priority group. If all the nodes in the highest priority group go **down** or hit a connection limit maximum, the BIG-IP Controller begins to pass connections on to nodes in the next lower priority group.

For example, in a configuration that has three priority groups, connections are first distributed to all nodes set as priority 3. If all priority 3 nodes are down, connections begin to be distributed to priority 2 nodes. If both the priority 3 nodes and the priority 2 nodes are down, connections then begin to be distributed to priority 1 nodes, and so on. Note, however, that the BIG-IP Controller continuously monitors the higher priority nodes, and each time a higher priority node becomes available, the BIG-IP Controller passes the next connection to that node.

Setting the global load balancing mode

The global load balancing mode is a system property of the BIG-IP Controller, and it applies to any pool with an **appgen_** name prefix.

To set the global load balancing mode in the Configuration utility

1. In the navigation pane, click **Virtual Servers**.
   The Virtual Servers screen opens.

2. In the **Node List Load Balance Method** box, choose the desired load balancing mode.
3. Click **Apply**.

WARNING

If you select Ratio mode or Priority mode, be sure to set the ratio weight or priority level for each node address in the configuration.

To set the load balancing mode on the command line

The command syntax for setting the load balancing mode is:

```
bigpipe lb <mode name>
```

Table 1.9 describes the valid options for the **<mode name>** parameter.

Mode Name	Description
`round robin`	Sets the load balancing mode to Round Robin mode.
`ratio`	Sets the load balancing mode to Ratio mode.
`priority`	Sets load balancing to Priority mode.
`least_conn`	Sets load balancing to Least Connections mode.
`fastest`	Sets load balancing to Fastest mode.
`observed`	Sets load balancing to Observed mode.
`predictive`	Sets load balancing to Predictive mode.

Table 1.9 *Options for the* ***<mode name>*** *parameter.*

Setting ratio weights and priority levels for node addresses

If you set the load balancing mode to either Ratio mode or Priority mode, you need to set a special property on each node address.

- **Ratio weight**
  The ratio weight is the proportion of total connections that the node address should receive. The default ratio weight for a given node address is 1. If all node addresses use this default weight, the connections are distributed equally among the nodes.
- **Priority level**
  The priority level assigns the node address to a specific priority group.

To set ratio weights and priority levels in the Configuration utility

1. In the navigation pane, click **Nodes**.
2. In the Nodes list, click the node for which you want to set the ratio weight.
   The Node Properties screen opens.
3. In the Node Properties screen, click the **Address** of the node.
   The Global Node Address Properties screen opens.
4. In the **Ratio or Priority** box, type the ratio weight of your choice.
5. Click the **Apply** button to save your changes.

To set ratio weights from the command line

The **bigpipe ratio** command sets the ratio weight for one or more node addresses:

```
bigpipe ratio <node IP> [<node IP>...] <ratio weight>
```

The following example defines ratio weights and priority for three node addresses. The first command sets the first node to receive half of the connection load. The second command sets the two remaining node addresses to each receive one quarter of the connection load.

```
bigpipe ratio 192.168.10.01  2
bigpipe ratio 192.168.10.02  192.168.10.03  1
```

WARNING

If you set the load balancing mode to Ratio or Priority, you must define the ratio or priority settings for each node address. The value you define using the ***bigpipe ratio*** *command is used as the ratio value if Ratio is the currently selected load balancing mode, and the same value is used as the priority level if Priority is the currently selected load balancing mode.*

Setting the load balancing method for a pool

This example describes how to change the load balancing method for a pool to use Ratio load balancing. For information about the other load balancing methods you can use to load balance a pool, see *Pool*, on page 1-62.

If you want to switch the load balancing method used in a pool from Round Robin to Ratio you must modify the pool specification in the Configuration utility or from the command line. You change the load balancing mode to **ratio_member**, and you must assign a ratio weight to each member of the pool.

Switching to Ratio mode

First, you should set the load balancing mode to Ratio. The load balancing mode is actually a property of the BIG-IP Controller system, and it applies to all virtual servers defined on the system.

To switch the system to Ratio mode in the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the toolbar, click the **Add Pool** button.
   The Add Pool screen opens.
3. In the **Pool Name** box, type in the name you want to use for the pool.

4. Click on the load balancing mode list and select **Ratio (member)**.
5. Use the resources options to set the **Ratio** value for the members in the pool. In the **Current Members** list, click the member you want to edit. Click the back button (<<) to pull the member into the resources section. Change the Ratio value for the member.

   Ratio
   Type in a number to assign a ratio to this node within the pool. For example, if you are using the ratio load balancing mode and you type a **1** in this box, the node will have a lower priority in the load-balancing pool than a node marked **2**.
6. Click the add button (>>) to add the member back to the **Current Members** list.
7. Repeat steps 5 and 6 until you have set the ratio values for each member to your satisfaction.
8. Click the **Apply** button.

To switch the pool to Ratio mode on the command line

To switch the pool use the **modify** keyword with the **bigpipe pool** command. For example, if you want change the pool **my_pool**, to use the **ratio_member** load balancing mode, you can type the following command:

```
bigpipe pool my_pool modify { lb_mode ratio_member member
   11.12.1.101:80 ratio 1 priority 1 member 11.12.1.100:80 ratio 3
   priority 1 }
```

NAT

A network translation address provides a routable alias IP address that a node can use as its source IP address when making or receiving connections to clients on the external network. You can configure a unique NAT for each node address included in a virtual server mapping.

◆ Note

Note that NATs do not support port translation, and are not appropriate for FTP. You cannot define a NAT if you configure a default SNAT.

The attributes you can configure for a NAT are in Table 1.10

NAT Attributes	Description
Original address	The original address is the node IP address of a host that you want to be able to connect to through the NAT.
Translated address	The translated address is an IP address that is routable on the external network of the BIG-IP Controller. This IP address is the NAT address.
Interface name	You can specify an interface for the NAT if you have more than one internal interface configured in the BIG-IP Controller.
Unit ID	You can specify a unit ID for a NAT if the BIG-IP Controller is configured to run in active active mode.

Table 1.10 *The attributes you can configure for a NAT.*

The IP addresses that identify nodes on the BIG-IP Controller's internal network need not be routable on the external network. This protects nodes from illegal connection attempts, but it also prevents nodes (and other hosts on the internal network) from receiving direct administrative connections, or from initiating connections to clients, such as mail servers or databases, on the BIG-IP Controller's external interface (destination processing).

Using network address translation resolves this problem. Network address translations (NATs) assign to a particular node a routable IP address that the node can use as its source IP address when connecting to servers on the BIG-IP Controller's external interface. You can use the NAT IP address to connect directly to the node through the BIG-IP Controller, rather than having the BIG-IP Controller send you to a random node according to the load balancing mode. IP forwarding provides functionality similar to a NAT. If your network does not support NATs, you may want to consider using IP forwarding.

Note

In addition to these options, you can set up forwarding virtual servers which allow you to selectively forward traffic to specific addresses. The BIG-IP Controller maintains statistics for forwarding virtual servers.

A network translation address provides a routable alias IP address that a node can use as its source IP address when making or receiving connections to clients on the external network. You can configure a unique NAT for each node address included in a virtual server mapping.

Note that NATs do not support port translation, and are not appropriate for FTP.

Note

You cannot define a NAT if you define a default SNAT.

WARNING

NATs do not support the NT Domain or CORBA protocols. Instead of using NATs, you need to configure IP forwarding (see Setting up IP forwarding, on page 1-30).

Defining a network address translation (NAT)

When you define standard network address translations (NATs), you need to create a separate NAT for each node that requires a NAT. You also need to use unique IP addresses for NAT addresses; a NAT IP address cannot match an IP address used by any virtual or physical servers in your network. You can configure a NAT with the Configuration utility or from the command line.

To configure a NAT in the Configuration utility

1. In the navigation pane, click **NATs**.
   The Network Address Translations screen opens.
2. On the toolbar, click **Add NAT**.
   The Add Nat screen opens.
3. In the **Node Address** box, type the IP address of the node.
4. In the **NAT Address** box, type the IP address that you want to use as the node's alias IP address.
5. In the **NAT Netmask** box, type an optional netmask. If you leave this box blank, the BIG-IP Controller generates a default broadcast address based on the IP address and netmask of this virtual server.
6. In the **NAT Broadcast** box, type the broadcast address. If you leave this box blank, the BIG-IP Controller generates a default broadcast address based on the IP address and netmask of this NAT.
7. In the **Interface** box, you can select an external interface (destination processing) on which the NAT address is to be used. Note that this setting only applies if the BIG-IP Controller has more than one external interface.
8. Click the **Apply** button.

To configure a NAT on the command line

A NAT definition maps the IP address of a node <**orig_addr**> to a routable address on the external interface <**trans_addr**>, and can include an optional interface and netmask specification. Use the following syntax to define a NAT:

```
bigpipe nat <orig_addr> to <trans_addr>[/<bitmask>] [<ifname>]
   [unit <unit ID>]
```

The <**ifname**> parameter is the internal interface of the BIG-IP Controller through which packets must pass to get to the destination internal address. The BIG-IP Controller can determine the interface to configure for the NAT in most cases. The <**ifname**> parameter is useful, for example, where there is more than one internal interface. You can use the **unit <unit ID>** parameter to specify the controller to which this NAT applies in an active-active redundant system.

The following example shows a NAT definition:

```
bigpipe nat 10.10.10.10 to 10.12.10.10/24 exp1
```

Deleting NATs

Use the following syntax to delete one or more NATs from the system:

```
bigpipe nat <orig_addr> [...<orig_addr>] delete
```

Displaying status of NATs

Use the following command to display the status of all NATs included in the configuration:

```
bigpipe nat show
```

Use the following syntax to display the status of one or more selected NATs (see Figure 1.3):

```
bigpipe nat <orig_addr> [...<orig_addr>] show
```

```
NAT { 10.10.10.3 to 9.9.9.9 }
 (pckts,bits) in = (0, 0), out = (0, 0)
NAT { 10.10.10.4 to 12.12.12.12
netmask 255.255.255.0 broadcast 12.12.12.255 }
 (pckts,bits) in = (0, 0), out = (0, 0)
```

Figure 1.3 *Output when you display the status of a NAT.*

Resetting statistics for a NAT

Use the following command to reset the statistics for an individual NAT:

```
bigpipe nat [<orig_addr>] stats reset
```

Use the following command to reset the statistics for all NATs:

```
bigpipe nat stats reset
```

Additional Restrictions

The **nat** command has the following additional restrictions:

- The IP address defined in the **<orig_addr>** parameter must be routable to a specific server behind the BIG-IP Controller.
- You must delete a NAT before you can redefine it.
- The interface for a NAT may only be configured when the NAT is first defined.

Node

Nodes are the network devices to which the BIG-IP Controller passes traffic. A node can be referenced by a load balancing pool. You can display information about nodes and set properties for nodes.

The attributes you can configure for a node are in Table 1.11

Node Attributes	Description
Enable/Disable nodes	You can enable or disable nodes independently from a load balancing pool.
Add a node as a member of a pool	You can add a node to a pool as a member. This allows you to use the load balancing and persistence methods defined in the pool to control connections handled by the node.

Table 1.11 *The attributes you can configure for a node.*

Enabling and disabling nodes and node addresses

To enable a node address, use the **node** command with a node address and the **enable** option:

```
bigpipe node 192.168.21.1 enable
```

To disable a node address, use the **node** command with the **disable** option:

```
bigpipe node 192.168.21.1 disable
```

To enable one or more node addresses, use the **node** command with a node address and port, and the **enable** option:

```
bigpipe node 192.168.21.1:80 enable
```

To disable one or more node addresses, use the **node** command with **disable** option:

```
bigpipe node 192.168.21.1:80 disable
```

Marking nodes and node ports up or down

To mark a node address down, use the **node** command with a node address and the **down** option (Note that marking a node **down** prevents the node from accepting new connections. Existing connections are allowed to complete):

```
bigpipe node 192.168.21.1 down
```

To mark a node address up, use the **node** command with the **up** option:

```
bigpipe node 192.168.21.1 up
```

To mark a particular port down, use the **node** command with a node address and port, and the **down** option (Note that marking a port down prevents the port from accepting new connections. Existing connections are allowed to complete):

```
bigpipe node 192.168.21.1:80 down
```

To mark a particular port up, use the **node** command with **up** option:

```
bigpipe node 192.168.21.1:80 up
```

Setting connection limits for nodes

Use the following command to set the maximum number of concurrent connections allowed on a node:

```
bigpipe node <node ip>[:<port>][...<node ip>[:<port>]] \
    limit <max conn>
```

Note that to remove a connection limit, you also issue the preceding command, but set the <**max conn**> variable to **0** (zero). For example:

```
bigpipe node 192.168.21.1:80 limit 0
```

Setting connection limits for node addresses

The following example shows how to set the maximum number of concurrent connections to **100** for a list of node addresses:

```
bigpipe node 192.168.21.1 192.168.21.1
    192.168.21.1 limit 100
```

To remove a connection limit, you also issue this command, but set the <**max conn**> variable to **0** (zero).

Displaying status of all nodes

When you issue the **node show** command, the BIG-IP Controller displays the node status (**up** or **down**, or **unchecked**), and a node summary of connection statistics, which is further broken down to show statistics by port.

`bigpipe node show`

The report shows the following information:

- current number of connections
- total number of connections made to the node since last boot
- maximum number of concurrent connections since the last boot
- concurrent connection limit on the node
- the total number of connections made to the node since last boot
- total number of inbound and outbound packets and bits

Figure 1.4 shows the output of this command:

```
bigpipe node 192.168.200.50:20
NODE 192.168.200.50    UP
|    (cur, max, limit, tot) = (0, 0, 0, 0)
|    (pckts,bits) in = (0, 0), out = (0, 0)
+-   PORT 20           UP
     (cur, max, limit, tot) = (0, 0, 0, 0)
     (pckts,bits) in = (0, 0), out = (0, 0)
```

Figure 1.4 *Node status and statistics*

Displaying the status of individual nodes and node addresses

Use the following command to display status and statistical information for one or more node addresses:

```
bigpipe node 192.168.21.1 show
```

The command reads the status of each node address, the number of current connections, total connections, and connections allowed, and the number of cumulative packets and bits sent and received.

Use the following command to display status and statistical information for one or more specific nodes:

```
bigpipe node 192.168.21.1:80 show
```

Resetting statistics for a node

Use the following command to reset the statistics for an individual node address:

```
bigpipe node [<node ip>:<port>] stats reset
```

Adding a node as a member to a pool

You can add a node as a member to a load balancing pool. For detailed information about how to do this, see *Pool*, on page 1-62.

Pool

Use the pool command to create, delete, modify, or display the pool definitions on the BIG-IP Controller. Use pools to group members together with a common load balancing mode and persistence mode.

Table 1.12 contains the attributes you can configure for a pool.

Pool Attributes	Description
Pool name	You can define the name of the pool.
Member specification	You can define each network device, or node, that is a member of the pool.
Load balancing method	You must define a specific load balancing method for a pool. You can have various pools configured with different load balancing methods.
Persistence method	You can define a specific persistence method for a pool. You can have various pools configured with different persistence methods.

Table 1.12 *The attributes of a pool.*

You can define pools from the command line, or define one in the web-based Configuration utility. This section describes how to define a simple pool using each of these configuration methods.

To create a pool using the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the toolbar, click the **Add Pool** button.
   The Add Pool screen opens.
3. In the **Pool Name** box, type in the name you want to use for the pool.
4. Click the load balancing mode list and select the load balancing mode you want to use for this pool.

5. Use the resources options to add members to the pool. To add a member to the pool, type the IP address in the **Node Address** box, type the port number in the **Port** box, and then type in the ratio or priority for this node. Finally, to add the node to the list, click the add (>>**)** button.

   - **Node Address**
     Type in the IP address of the node you want to add to the pool.
   - **Port**
     Type in the port number of the port you want to use for this node in the pool.
   - **Ratio**
     Type in a number to assign a ratio to this node within the pool. For example, if you are using the ratio load balancing mode and you type a **1** in this box, the node will have a lower priority in the load-balancing pool than a node marked **2**.
   - **Priority**
     Type in a number to assign a priority to this node within the pool. For example, if you are using a priority load-balancing mode and you type a **1** in this box, the node will have a lower priority in the load-balancing pool than a node marked **2**.
   - **Current Members**
     This is a list of the nodes that are part of the load balancing pool.

6. Click the **Apply** button.

To define a pool from the command line

To define a pool from the command line, use the following syntax:

```
bigpipe pool <pool_name> {lb_method <lb_method> member
   <member_definition> ... member <member_definition>}
```

For example, if you want to create the pool **my_pool**, with two members using Round Robin **(rr)** load balancing, from the command line, you would type the following command:

```
bigpipe pool my_pool { lb_method rr member 11.12.1.101:80 member
   11.12.1.100:80 }
```

Command line options

Use the following elements to construct pools from the command line:

Pool Element	Description
Pool name	A string from 1 to 31 characters, for example: **new_pool**
Member definition	member <ip address>:<port> [ratio <value>] [priority <value>]
lb_method_specificaton	lb_method [rr | ratio | priority | fastest | least_conn | predictive | observed | ratio_member | priority_member | least_conn_member]
persist_mode_specification	persist_mode [cookie | simple | ssl | sticky]

Table 1.13 *The elements you can use to construct a pool.*

Deleting a pool

To delete a pool use the following syntax:

```
bigpipe pool <pool_name> delete
```

All references to a pool must be removed before a pool can be deleted.

Modifying pools

You can use the command line to add or delete members from a pool. You can also modify the load balancing mode for a pool from the command line. To add a new member to a pool use the following syntax:

```
bigpipe pool <pool_name> add { 1.2.3.2:telnet }
```

To delete a member from a pool use the following syntax:

```
bigpipe pool <pool_name> delete { 1.2.3.2:telnet }
```

Display pools

Use the following syntax to display all pools:

```
bigpipe pool show
```

Use the following syntax to display a specific pool:

```
bigpipe pool <pool_name> show
```

Setting up persistence for a pool

If you are setting up an e-commerce or other type of dynamic content site, you may need to configure persistence on the BIG-IP Controller. Whether you need to configure persistence or not simply depends on how you store client-specific information, such as items in a shopping cart, or airline ticket reservations. For example, you may store the airline ticket reservation information in a back-end database that all nodes can access; or on the specific node to which the client originally connected; or in a cookie on the client's machine.

If you store client-specific information on specific nodes, you need to configure persistence. When you turn on persistence, returning clients can bypass load balancing and instead can go to the node where they last connected in order to get to their saved information.

The BIG-IP Controller tracks information about individual persistent connections, and keeps the information only for a given period of time. The way in which persistent connections are identified depends on the type of persistence. The BIG-IP Controller supports two basic types of persistence, and six advanced types of persistence. The two basic types of persistence are:

- **SSL persistence**
  SSL persistence is a type of persistence that tracks SSL connections using the SSL session ID, and it is a property of each individual pool. Using SSL persistence can be particularly important if your clients typically have translated IP addresses or dynamic IP addresses, such as those that Internet service

providers typically assign. Even when the client's IP address changes, the BIG-IP Controller still recognizes the connection as being persistent based on the session ID.

- **Simple persistence**
  Simple persistence supports TCP and UDP protocols, and it tracks connections based only on the client IP address. When a client requests a connection to a virtual server that supports simple persistence, the BIG-IP Controller checks to see if that client previously connected, and if so, returns the client to the same node.

You may want to use SSL persistence and simple persistence together. In situations where an SSL session ID times out, or where a returning client does not provide a session ID, you may want the BIG-IP Controller to direct the client to the original node based on the client's IP address. As long as the client's simple persistence record has not timed out, the BIG-IP Controller can successfully return the client to the appropriate node.

In addition to the simple persistence and SSL persistence options provided by the BIG-IP Controller, there are six advanced persistence options are available. The advanced options include:

- HTTP cookie persistence
- Destination address affinity (sticky persistence)
- Persist masking
- Maintaining persistence across virtual servers with the same address
- Maintaining persistence across all virtual servers
- Backward compatibility with node list virtual servers

Note

All persistence methods are properties of pools

Setting up SSL persistence

SSL persistence is a property of a pool. You can set up SSL persistence from the command line or from the Configuration utility. To set up SSL persistence, you need to do two things:

- Turn SSL persistence on.
- Set the SSL session ID timeout, which determines how long the BIG-IP Controller stores a given SSL session ID before removing it from the system.

To configure SSL persistence using the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. Click the appropriate pool in the list.
   The Pool Properties screen opens.
3. In the toolbar, click the **Persistence** button.
   The Pool Persistence screen opens.
4. Click the **SSL Persistence** button.
5. In the **Timeout** box, type the number of seconds that the BIG-IP Controller should store SSL sessions IDs before removing them from the system.
6. Click the **Apply** button.

To activate SSL persistence from the command line

Use the following syntax to activate SSL persistence from the command line:

```
bigpipe pool <pool_name> modify { persist_mode ssl ssl_timeout
   <timeout> simple_mask <ip_mask> }
```

For example, if you want to set SSL persistence on the pool **my_pool**, type the following command:

```
bigpipe pool my_pool modify { persist_mode ssl ssl_timeout 3600
   simple_mask 255.255.255.0 }
```

Display persistence information for a pool

To show the persistence configuration for the pool:

```
bigpipe pool <pool_name> persist show
```

To display all persistence information for the pool named **classc_pool**, use the **show** option:

```
bigpipe pool classc_pool persist show
```

Setting up simple persistence

You can set simple persistence properties for both an individual virtual server, and for a port. Individual virtual server persistence settings can override those of the port. When you set simple persistence on a port, all virtual servers that use the given port inherit the port's persistence settings.

Setting simple persistence on virtual servers

Persistence settings for pools apply to both TCP and UDP persistence. When the persistence timer is set to a value greater than 0, persistence is **on**. When the persistence timer is set to 0, persistence is **off**.

To configure simple persistence for pools using the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. Select the pool for which you want to configure simple persistence.
   The Pool Properties screen opens.
3. In the toolbar, click the **Persistence** button.
   The Pool Persistence Properties screen opens.
4. In the Persistence Type section, click the **Simple Persistence** button.
   Type the following information:

- **Timeout (seconds)**
  Set the number of seconds for persistence on the pool. (This option is not available if you are using rules.)
- **Mask**
  Set the persistence mask for the pool. The persistence mask determines persistence based on the portion of the client's IP address that is specified in the mask.

5. Click the **Apply** button.

To configure simple persistence for pools from the command line

You can use the **bigpipe pool** command with the **modify** keyword sets simple persistence for a pool. Note that a timeout greater than 0 turns persistence **on**, and a timeout of 0 turns persistence **off**.

```
bigpipe pool <pool_name> modify { persist_mode ssl ssl_timeout
   <timeout> simple_mask <ip_mask> }
```

For example, if you want to set SSL persistence on the pool **my_pool**, type the following command:

```
bigpipe pool my_pool modify { persist_mode ssl
ssl_timeout 3600 simple_mask 255.255.255.0 }
```

Using HTTP cookie persistence

You can set up the BIG-IP Controller to use HTTP cookie persistence. This method of persistence uses an HTTP cookie stored on a client's computer to allow the client to reconnect to the same server previously visited at a web site.

There are four types of cookie persistence available:

❖ Insert mode

❖ Rewrite mode

❖ Passive mode

❖ Hash mode

The mode you choose affects how the cookie is handled by the BIG-IP Controller when it is returned to the client.

Insert mode

If you specify Insert mode, the information about the server to which the client connects is inserted in the header of the HTTP response from the server as a cookie. The cookie is named **BIGipServer <pool_name>**, and it includes the address and port of the server handling the connection. The expiration date for the cookie is set based on the timeout configured on the BIG-IP Controller.

To activate Insert mode in the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the Pools list, click the pool for which you want to set up Insert mode.
   The properties screen for the pool you clicked opens.
3. In the toolbar, click the **Persistence** button.
   The Pool Persistence screen opens.
4. Click the **Active HTTP Cookie** button.
5. Select Insert mode from the Method list.
6. Type the timeout value in days, hours, minutes, and seconds. This value determines how long the cookie lives on the client computer before it expires.
7. Click the **Apply** button.

To activate Insert HTTP cookie persistence from the command line

To activate Insert mode from the command line, use the following syntax:

```
bigpipe pool <pool_name> { <lb_mode_specification> persist_mode
   cookie cookie_mode insert cookie_expiration <timeout> <member
   definition> }
```

The **<timeout>** value for the cookie is written using the following format:

```
<days>d hh:mm:ss
```

Rewrite mode

If you specify Rewrite mode, the BIG-IP Controller intercepts a Set-Cookie, named **BIGipCookie**, sent from the server to the client and overwrites the name and value of the cookie. The new cookie is named **BIGipServer <pool_name>** and it includes the address and port of the server handling the connection.

Rewrite mode requires you to set up the cookie created by the server. In order for Rewrite mode to work, there needs to be a blank cookie coming from the web server for the BIG-IP Controller to rewrite. With Apache variants, the cookie can be added to every web page header by adding an entry in the **httpd.conf** file:

```
Header add Set-Cookie
   BIGipCookie=00000000000000000000000000...
```

(The cookie may contain a total of 120 zeros.)

WARNING

For backward compatibility the blank cookie can contain only 75 zeros. However, cookies of this size do not allow you to use rules and persistence together.

To activate Rewrite mode cookie persistence in the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the **Pools** list, click the pool for which you want to set up Rewrite mode.
   The properties screen for the pool you clicked opens.
3. In the toolbar, click the **Persistence** button.
   The Pool Persistence screen opens.
4. Click the **Active HTTP Cookie** button.
5. Select Rewrite mode from the **Method** list.
6. Type the timeout value in days, hours, minutes, and seconds. This value determines how long the cookie lives on the client computer before it expires.

7. Click the **Apply** button.

To activate Rewrite mode cookie persistence from the command line

To activate Rewrite mode from the command line, use the following syntax:

```
bigpipe pool <pool_name> { <lb_mode_specification> persist_mode
   cookie cookie_mode rewrite cookie_expiration <timeout> <member
   definition> }
```

The **<timeout>** value for the cookie is written using the following format:

```
<days>d hh:mm:ss
```

Passive mode

If you specify Passive mode, the BIG-IP Controller does not insert or search for blank Set-Cookies in the response from the server. It does not try to set up the cookie. In this mode, it is assumed that the server provides the cookie formatted with the correct node information and timeout.

In order for Passive mode to work, there needs to be a cookie coming from the web server with the appropriate node information in the cookie. With Apache variants, the cookie can be added to every web page header by adding an entry in the **httpd.conf** file:

```
Header add Set-Cookie: "BIGipServer my_pool=184658624.20480.000;
   expires=Sat, 19-Aug-2000 19:35:45 GMT; path=/"
```

In this example, **my_pool** is the name of the pool that contains the server node, **184658624** is the encoded node address and **20480** is the encoded port.

The equation for an address (a.b.c.d) is:

d*256^3 + c*256^2 + b*256 +a

The way to encode the port is to take the two bytes that store the port and reverse them. So, port 80 becomes 80 * 256 + 0 = 20480. Port 1433 (instead of 5 * 256 + 153) becomes 153 * 256 + 5 = 39173.

To activate Passive mode cookie persistence in the Configuration utility

After you set up the cookie created by the web server, you must activate Passive mode on the BIG-IP Controller.

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the Pools list, click the pool for which you want to set up Passive mode.
   The properties screen for the pool you clicked opens.
3. In the toolbar, click the **Persistence** button.
   The Pool Persistence screen opens.
4. Select Passive HTTP Cookie mode.
5. Click the **Apply** button.

To activate Passive mode cookie persistence from the command line

After you set up the cookie created by the web server, you must activate Passive mode on the BIG-IP Controller. To activate HTTP cookie persistence from the command line, use the following syntax:

```
bigpipe pool <pool_name> { <lb_mode_specification> persist_mode
   cookie cookie_mode passive <member definition> }
```

Note

The ***<timeout>*** *value is not used in Passive mode.*

Hash mode

If you specify Hash mode, the hash mode consistently maps a cookie value to a specific node. When the client returns to the site, the BIG-IP Controller uses the cookie information to return the client to a given node. With this mode, the web server must generate the cookie. The BIG-IP Controller does not create the cookie automatically like it does with Insert mode.

To configure the cookie persistence hash option in the Configuration utility

Before you follow this procedure, you must configure at least one pool.

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the Pools list, click the pool for which you want to set up hash mode persistence.
   The properties screen for the pool you clicked opens.
3. In the toolbar, click the **Persistence** button.
   The Pool Persistence screen opens.
4. Click the **Cookie Hash** button.
   Set the following values (see Table 1.14 for more information):
   - **Cookie Name**
     Type in the name of an HTTP cookie being set by the Web site. This could be something like Apache or SSLSESSIONID. It depends on the type of web server your site is running.
   - **Hash Values**
     The **Offset** is the number of bytes in the cookie to skip before calculating the hash value. The **Length** is the number of bytes to use when calculating the hash value.
5. Click the **Apply** button.

To configure the hash cookie persistence option from the command line

Use the following syntax to configure the hash cookie persistence option:

```
bigpipe pool <pool_name> { <lb_mode_specification> persist_mode
   cookie cookie_mode hash cookie_hash_name <cookie_name>
   cookie_hash_offset <cookie_value_offset> cookie_hash_length
   <cookie_value_length> <member definition> }
```

The **<cookie_name>**, **<cookie_value_offset>**, and **<cookie_value_length>** values are described in Table 1.14:

Hash mode values	Description
`<cookie_name>`	This is the name of an HTTP cookie being set by a Web site.
`<cookie_value_offset>`	This is the number of bytes in the cookie to skip before calculating the hash value.
`<cookie_value_length>`	This is the number of bytes to use when calculating the hash value.

Table 1.14 *The cookie hash mode values*

Using destination address affinity (sticky persistence)

You can optimize your proxy server array with destination address affinity (also called sticky persistence). Address affinity directs requests for a certain destination to the same proxy server, regardless of which client the request comes from.

This enhancement provides the most benefits when load balancing caching proxy servers. A caching proxy server intercepts web requests and returns a cached web page if it is available. In order to improve the efficiency of the cache on these proxies, it is necessary to send similar requests to the same proxy server repeatedly. Destination address affinity can be used to cache a given web page on one proxy server instead of on every proxy server in an array. This saves the other proxies from having to duplicate the web page in their cache, wasting memory.

WARNING

In order to prevent sticky entries from clumping on one server, use a static load balancing mode for the members of the pool, such as Round Robin.

To activate destination address affinity in the Configuration utility

You can only activate destination address affinity on pools directly or indirectly referenced by wildcard virtual servers. For information on setting up a wildcard virtual server, see the ***Administrator Guide***, *Defining wildcard virtual servers*. Follow these steps to configure destination address affinity:

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the Pools list, click the pool for which you want to set up destination address affinity.
   The properties screen for the pool you clicked opens.
3. In the toolbar, click the **Persistence** button.
   The Pool Persistence screen opens.
4. Click the **Destination Address Affinity** button to enable destination address affinity.
5. In the **Mask** box, type in the mask you want to apply to sticky persistence entries.
6. Click the **Apply** button.

To activate sticky persistence from the command line

Use the following command to enable sticky persistence for a pool:

```
bigpipe pool <pool_name> modify { persist_mode sticky <enable |
   disable> sticky_mask <ip address> }
```

Use the following command to disable sticky persistence for a pool:

```
bigpipe pool <pool_name> modify { persist_mode sticky disable
   sticky_mask <ip address> }
```

Use the following command to delete sticky entries for the specified pool:

```
bigpipe pool <pool_name> sticky clear
```

To show the persistence configuration for the pool:

```
bigpipe pool <pool_name> persist show
```

Using a simple timeout and a persist mask on a pool

The persist mask feature works only on pools that implement simple persistence. By adding a persist mask, you identify a range of client IP addresses to manage together as a single simple persistent connection when connecting to the pool.

To apply a simple timeout and persist mask in the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the Pools list, click the pool for which you want to set up simple persistence.
   The properties screen for the pool you clicked opens.
3. In the toolbar, click the **Persistence** button.
   The Pool Persistence screen opens.
4. Select Simple Persistence mode.
5. In the **Timeout** box, type the timeout in seconds.
6. In the **Mask** box, type the persist mask you want to apply.
7. Click the **Apply** button.

To apply a simple timeout and persist mask from the command line

The complete syntax for the command is:

```
bigpipe pool <pool_name> modify { [<lb_mode_specification>]
   persist_mode simple simple_timeout <timeout> simple_mask
   <dot_notation_longword> }
```

For example, the following command would keep persistence information together for all clients within a C class network that connect to the pool **classc_pool**:

```
bigpipe pool classc_pool modify { persist_mode simple
   simple_timeout 1200 simple_mask 255.255.255.0 }
```

You can turn off a persist mask for a pool by using the **none** option in place of the **simple_mask** mask. To turn off the persist mask that you set in the preceding example, use the following command:

```
bigpipe pool classc_pool modify { simple_mask none }
```

To display all persistence information for the pool named **classc_pool**, use the **show** option:

```
bigpipe pool classc_pool persist show
```

Maintaining persistence across virtual servers that use the same virtual addresses

When this mode is turned on, the BIG-IP Controller attempts to send all persistent connection requests received from the same client, within the persistence time limit, to the same node only when the virtual server hosting the connection has the same virtual address as the virtual server hosting the initial persistent connection. Connection requests from the client that go to other virtual servers with different virtual addresses, or those connection requests that do not use persistence, are load balanced according to the load balancing mode defined for the pool.

If a BIG-IP Controller configuration includes the following virtual server mappings, where the virtual server **v1:http** references the **http_pool** (contains the nodes **n1:http** and **n2:http**) and the virtual server **v1:ssl** references the pool **ssl_pool** (contains the nodes **n1:ssl** and **n2:ssl**). Each virtual server uses persistence:

```
bigpipe vip v1:http use pool http_pool
bigpipe vip v1:ssl use pool ssl_pool
bigpipe vip v2:ssl use pool ssl_pool
```

For example, a client makes an initial connection to **v1:http** and the load balancing mechanism assigned to the pool **http_pool** chooses **n1:http** as the node. If the same client then connects to **v2:ssl**, the BIG-IP Controller starts tracking a new persistence session, and it uses the load balancing mode to determine which node should receive the connection request because the requested virtual server uses a different virtual address (**v2**) than the virtual server hosting the first persistent connection request (**v1**). However, if the client

subsequently connects to **v1:ssl**, the BIG-IP Controller uses the persistence session established with the first connection to determine the node that should receive the connection request, rather than the load balancing mode. The BIG-IP Controller should send the third connection request to **n1:ssl**, which uses the same node address as the **n1:http** node that currently hosts the client's first connection with which it shares a persistent session.

WARNING

In order for this mode to be effective, virtual servers that use the same virtual address, as well as those that use TCP or SSL persistence, should include the same node addresses in the virtual server mappings.

The system control variable **bigip.persist_on_any_port_same_vip** turns this mode on and off. To activate the persistence mode, type:

```
sysctl -w bigip.persist_on_any_port_same_vip=1
```

To deactivate the persistence mode, type:

```
sysctl -w bigip.persist_on_any_port_same_vip=0
```

To activate persistence for virtual servers that use the same address in the Configuration utility

1. In the navigation pane, click the BIG-IP Controller icon. The BIG-IP System Properties screen opens.
2. In the toolbar, click the **Advanced Properties** button. The BIG-IP System Control Variables screen opens.
3. Click the **Allow Persistence Across All Ports for Each Virtual Address** checkbox to activate this persistence mode. Clear the checkbox to disable this persistence mode.
4. Click the **Apply** button.

Maintaining persistence across all virtual servers

You can set the BIG-IP Controller to maintain persistence for all connections requested by the same client, regardless of which virtual server hosts each individual connection initiated by the

client. When this mode is turned on, the BIG-IP Controller attempts to send all persistent connection requests received from the same client, within the persistence time limit, to the same node. Connection requests from the client that do not use persistence are load balanced according to the currently selected load balancing mode.

If a BIG-IP Controller configuration includes the following virtual server mappings, where the virtual servers **v1:http** and **v2:http** reference the **http1_pool** and **http2_pool** (both pools contain the nodes **n1:http** and **n2:http**) and the virtual servers **v1:ssl** and **v2:ssl** reference the pools **ssl1_pool** and **ssl2_pool** (both pools contain the nodes **n1:ssl** and **n2:ssl**). Each virtual server uses persistence:

```
bigpipe vip v1:http use pool http1_pool
bigpipe vip v1:ssl use pool ssl1_pool
bigpipe vip v2:http use pool http2_pool
bigpipe vip v2:ssl use pool ssl2_pool
```

Say that a client makes an initial connection to **`v1:http`** and the BIG-IP Controller's load balancing mechanism chooses **`n1:http`** as the node. If the same client subsequently connects to **`v1:ssl`**, the BIG-IP Controller would send the client's request to **`n1:ssl`**, which uses the same node address as the **`n1:http`** node that currently hosts the client's initial connection. What makes this mode different from maintaining persistence across virtual servers that use the same virtual address is that if the same client subsequently connects to **`v2:ssl`**, the BIG-IP Controller would send the client's request to **`n1:ssl`**, which uses the same node address as the **`n1:http`** node that currently hosts the client's initial connection.

WARNING

In order for this mode to be effective, virtual servers that use TCP or SSL persistence should include the same member addresses in the virtual server mappings.

The system control variable **bigip.persist_on_any_vip** turns this mode **on** and **off**. To activate the persistence mode, type:

```
sysctl -w bigip.persist_on_any_vip=1
```

To deactivate the persistence mode, type:

```
sysctl -w bigip.persist_on_any_vip=0
```

To activate persistence across all virtual servers in the Configuration utility

1. In the navigation pane, click the BIG-IP Controller icon. The BIG-IP System Properties screen opens.
2. In the toolbar, click the **Advanced Properties** button. The BIG-IP System Control Variables screen opens.
3. Click the **Allow Persistence Across All Virtual Servers** checkbox to activate this persistence mode. Clear the checkbox to disable this persistence mode.
4. Click the **Apply** button.

Backward compatible persistence for node list virtual servers

It is still possible to configure persistence by virtual server and port. You must configure virtual servers that reference a pool or a rule by modifying the pool.

Virtual server definitions containing a node list and persistence settings are converted into an independent pool with the name **appgen_<virtual_addr>.<virtual_port>** and a virtual server that references the pool. The pool persistence settings are set to mimic the behavior of a virtual server with persistence. For example, the nodelist virtual server definition.

```
vip 168.1.1.1:80 { define 10.1.1.1:80 10.2.2.2:80
    special cookie rewrite 10d }
```

This virtual server definition is stored and written in the **/etc/bigip.conf** file in the following manner:

```
pool  appgen_168.1.1.1.80  {
    lb_mode round_robin
    persist_mode cookie
    cookie_mode rewrite
    cookie_expiration 10d
    member 10.1.1.1:80
    member 10.2.2.2:80
}
vip 168.1.1.1:80 { use pool appgen_168.1.1.1.80 }
```

Figure 1.5 *An example of an* ***appgen_pool*** *created from a node list virtual server*

While you can still apply vitual port simple persistence timeouts they are not saved a part of the BIG-IP Controller configuration. Defining a virtual port timeout affects the persistence configuration of pools that are directly referenced by virtual servers with a matching virtual port. When a virtual port timeout is defined, pools with a persistence mode of **none** are changed to **simple**, and the simple persistence timeouts are changed from **0** to the virtual port timeout.

The virtual server **simple** and **sticky** persistence commands operate on the pool referenced by the virtual server instead of on the virtual server itself. You cannot use commands to display information for a virtual server that does not reference a pool. Virtual server persistence modifications are:

vip <ip>:<port> persist <value>

vip <ip>:<port> persist mask <ip>

vip <ip>:<port> sticky (enable | disable | clear)

vip <ip>:<port> sticky mask <ip>

vip <ip>:<port> mirror persist (enable | disable)

All virtual server persistence queries now return error messages and a suggested pool persistence query. Virtual server persistence queries that now generate errors are:

vip <ip>:<port> persist (show | dump)

```
vip <ip>:<port> persist mask show
vip <ip>:<port> sticky (show | dump)
vip <ip>:<port> sticky mask show
vip <ip>:<port> mirror persist show
```

Port

One of the security features of the BIG-IP Controller is that all ports on the controller are locked down and unavailable for service unless you specifically open them to network access. Before clients can use the virtual servers you have defined, you must allow access to each port that the virtual servers use.

◆ Tip

Virtual servers using the same service actually share a port on the BIG-IP Controller. This command is global, you only need to open access to a port once; you do not need to open access to a port for each instance of a virtual server that uses it.

A port is any valid port number, between 0 and 65535, inclusive, or any valid service name in the **/etc/services** file.

The attributes you can configure for a port are in Table 1.15

Attributes	Description
Allow access to ports	As a security measure, all ports are locked down on the BIG-IP Controller. In order for the BIG-IP Controller to load balance traffic, you must enable access to the port on which the BIG-IP Controller will receive traffic.
Connection limits	You can define a connection limit for a port so that a flood of connections does not overload the BIG-IP Controller.

Table 1.15 *The attributes you can configure for an port.*

To allow access to services in the Configuration utility

Any time you create a virtual server and define a port or service with the Configuration utility, the port or service is automatically enabled.

To allow access to services on the command line

Using the **bigpipe port** command, you can allow access to one or more ports at a time.

```
bigpipe port <port>... <port> enable
```

For example, in order to enable HTTP (port 80) and Telnet (port 23) services, you can enter the following **bigpipe port** command:

```
bigpipe port 80 23 443 enable
```

WARNING

*In order for FTP to function properly, you must allow both ports 20 and 21 (or **ftp-data** and **ftp**).*

Allowing and denying virtual ports

You can enable or disable traffic to specific virtual ports. The default setting for all virtual ports is disabled. Use the following syntax to allow one or more virtual ports:

```
bigpipe port <port> [...<port>] enable
```

To deny access to one or more virtual ports:

```
bigpipe port <port> [...<port>] disable
```

Setting connection limits on ports

Use the following syntax to set the maximum number of concurrent connections allowed on a virtual port. Note that you can configure this setting for one or more virtual ports.

```
bigpipe port <port> [...<port>] limit <max conn>
```

To turn off a connection limit for one or more ports, use the preceding command, setting the <**max conn**> parameter to **0** (zero):

```
bigpipe port <port> [...<port>] limit 0
```

Displaying the status of all virtual ports

Use the following syntax to display the status of virtual ports included in the configuration:

```
bigpipe port show
```

Displaying the status for specific virtual ports

Use the following syntax to display the status of one or more virtual ports:

bigpipe port <port> [...<port>] show

Figure 1.6 shows a sample of formatted output of the port command.

```
bigpipe port telnet show

PORT  23     telnet        enable
(cur, max, limit, tot, reaped) = (37,73,100,691,29)
      (pckts,bits) in = (2541, 2515600), out = (2331, 2731687)
```

Figure 1.6 *Formatted output of* ***port*** *command showing the Telnet port statistics*

Redundant System

Redundant BIG-IP Controller systems have special settings that you need to configure, such as interface fail-safe settings. One convenient aspect of configuring a redundant system is that once you have configured one of the controllers, you can simply copy the configuration to the other controller in the system using the configuration synchronization feature in the **bigpipe** command line tool or in the Configuration utility.

There are two basic aspects about working with redundant systems:

- Synchronizing configurations between two controllers
- Configuring fail-safe settings for the interfaces

In addition to the simple redundant features available on the BIG-IP Controller, several advanced redundant features are available. Advanced redundant system features provide additional assurance that your content is available if a BIG-IP Controller experiences a problem. These advanced redundant system options include:

- Mirroring connection and persistence information
- Gateway fail-safe
- Network-based fail-over
- Setting a specific BIG-IP Controller to be the active controller
- Setting up active-active redundant controllers

The attributes you can configure for a redundant systems are in Table 1.16

Attributes	Description
Synchronizing configurations	This feature allows you to configure one controller and then synchronize the configuration with the other controller.
Fail-safe for interfaces	Fail-safe for interfaces provides the ability to cause a controller to fail over if an interface is no longer generating traffic.
Mirroring connections and persistence information	You can mirror connection and/or persistence information between redundant controllers. This enables you to provide seamless fail-over of client connections.
Gateway fail-safe	This feature allows you to fail-over between two gateway routers.
Network-based fail-over	You can configure the BIG-IP Controller to use the network to determine the status of the active controller.
Setting a dominant controller	You can set up one controller in a pair to be the dominant active controller. The controller you set up as the dominant controller will always attempt to be active.
Active-active configuration	The default mode for a BIG-IP Controller redundant system is Active/Standby. However, you can configure both controllers to run in active mode.

Table 1.16 *The attributes you can configure for redundant systems.*

Preparing to use the synchronization command

Before you can use the **bigpipe configsync** command or the Configuration utility to synchronize domestic HA redundant BIG-IP Controllers, you must first run the **config_failover** command. This command performs the following tasks:

- Checks for a fail-over IP address for the other controller in BIG/db.
- Verifies that the **AllowHosts** entry in the **/etc/sshd_config** file includes the IP address of the other controller in the redundant configuration.
- Runs the **ssh-keygen** command which creates the security keys for the controller.

- Shares the security keys with the other controller in the redundant system.

To run the **config_failover** command, type the following command from the command line:

```
config_failover
```

The **config_failover** utility prompts you for the root password of the other controller in the redundant system before it generates the security keys for the BIG-IP Controller.

Synchronizing configurations between controllers

Once you complete the initial configuration on the first controller in the system, you can synchronize the configurations between the active unit and the standby unit. When you synchronize a configuration, the following configuration files are copied to the other BIG-IP Controller:

- **The common keys in BIG/db**
- **/etc/bigip.conf**
  The **/etc/bigip.conf** file stores virtual server and node definitions and settings, including node ping settings, the load balancing mode, and NAT and SNAT settings.
- **/etc/bigd.conf**
  The **/etc/bigd.conf** file stores service check settings.
- **/etc/hosts.allow**
  The **/etc/hosts.allow** file stores the IP addresses that are allowed to make administrative shell connections to the BIG-IP Controller.
- **/etc/hosts.deny**
  The **/etc/hosts.deny** file stores the IP addresses that are not allowed to make administrative shell connections to the BIG-IP Controller.
- **User account files**
- **/etc/ipfw.conf** and **/etc/ipfw.filt**
  The **/etc/ipfw.conf** and **/etc/ipfw.filt** files store IP filter settings.

- **rc.sysctl**
  The **rc. sysctl** file contains system control variable settings.
- **/etc/rateclass.conf**
  The **/etc/rateclass.conf** file stores rate class definitions.
- **/etc/ipfwrate.conf** and **/etc/ipfwrate.filt**
  The **/etc/ipfwrate.conf** and **/etc/ipfwrate.filt** files store IP filter settings for filters that also use rate classes.
- **/etc/snmpd.conf**
  The **/etc/snmpd.conf** file stores SNMP configuration settings.

If you use command line utilities to set configuration options, be sure to save the current configuration to the file before you use the configuration synchronization feature. Use the following **bigpipe** command to save the current configuration:

```
bigpipe -s
```

WARNING

If you are synchronizing with a controller that already has configuration information defined, we recommend that you back up that controller's original configuration file(s).

To synchronize the configuration using the Configuration utility

1. In the navigation pane, click the BIG-IP Controller icon.
   The BIG-IP System Properties screen opens.
2. On the toolbar, click the **Sync Configuration** button.
   The Sync Configuration screen opens.
3. Click the **Synchronize** button.

To synchronize the configuration from the command line

You use the **bigpipe configsync** command to synchronize configurations. When you include the **all** option in the command, all the configuration files are synchronized between machines.

```
bigpipe configsync all
```

If you want to synchronize only the **/etc/bigip.conf** file, you can use the same command without any options:

```
bigpipe configsync
```

Configuring fail-safe settings

For maximum reliability, the BIG-IP Controller supports failure detection on both internal and external interface cards. When you arm the fail-safe option on an interface card, the BIG-IP Controller monitors network traffic going through the interface. If the BIG-IP Controller detects a loss of traffic on an interface when half of the fail-safe timeout has elapsed, it attempts to generate traffic. An interface attempts to generate network traffic by issuing ARP requests to nodes accessible through the interface. Also, an ARP request is generated for the default route if the default router is accessible from the interface. Any traffic through the interface, including a response to the ARP requests, averts a fail-over.

If the BIG-IP Controller does not receive traffic on the interface before the timer expires, it initiates a fail-over, switches control to the standby unit, and reboots.

WARNING

You should arm the fail-safe option on an interface only after the BIG-IP Controller is in a stable production environment. Otherwise, routine network changes may cause fail-over unnecessarily.

Arming fail-safe on an interface

Each interface card installed on the BIG-IP Controller has a unique name, which you need to know when you set the fail-safe option on a particular interface card. You can view interface card names in the Configuration utility, or you can use the **bigpipe interface** command to display interface names on the command line.

To arm fail-safe on an interface using the Configuration utility

1. In the navigation pane, click **NICs** (network interface cards).
   The Network Interface Cards list opens and displays each installed NIC.
2. Select an interface name.
   The Network Interface Card Properties screen opens.
3. Check **Arm Failsafe** to turn on the fail-safe option for the selected interface.
4. In the **Timeout** box, type the maximum time allowed for a loss of network traffic before a fail-over occurs.
5. Click the **Apply** button.

To arm fail-safe on an interface from the command line

One of the required parameters for the **bigpipe interface** command is the name of the interface. If you need to look up the names of the installed interface cards, use the **bigpipe interface** command with the **show** keyword:

```
bigpipe interface show
```

To arm fail-safe on a particular interface, use the **bigpipe interface** command with the **failsafe arm** keyword and interface name parameter:

```
bigpipe interface <ifname> timeout <seconds>
bigpipe interface <ifname> failsafe arm
```

For example, you have an external interface named **exp0** and an internal interface named **exp1**. To arm the fail-safe option on both cards with a timeout of 30 seconds, you need to issue the following commands:

```
bigpipe interface exp0 timeout 30
bigpipe interface exp1 timeout 30
bigpipe interface exp0 failsafe arm
bigpipe interface exp1 failsafe arm
```

Mirroring connection and persistence information

When the fail-over process puts the active controller duties onto a standby controller, the connection capability of your site returns so quickly that it has little chance to be missed. By preparing a redundant system for the possibility of fail-over, you effectively maintain your site's reliability and availability in advance. But fail-over alone is not enough to preserve the connections and transactions on your servers at the moment of fail-over; they would be dropped as the active controller goes down unless you have enabled ***mirroring***.

The mirror feature on BIG-IP Controllers is a specialized, ongoing communication between the active and standby controllers that duplicates the active controller's real-time connection or persistence information state on the standby controller. If mirroring has been enabled, fail-over can be seamless to such an extent that file transfers can proceed uninterrupted, customers making orders can complete transactions without interruption, and your servers can generally continue with whatever they were doing at the time of fail-over.

The mirror feature is intended for use with long-lived connections, such as FTP, Chat, and Telnet sessions. Mirroring is also effective for persistence information.

WARNING

If you attempt to mirror all connections, the performance of the BIG-IP Controller may degrade.

Commands for mirroring

Table 1.17 contains the commands that support mirroring capabilities. For complete descriptions, syntax, and usage examples, see the ***BIG-IP Controller Reference Guide***, *BIG/pipe Command Reference*.

BIG/pipe command	Options
`bigpipe mirror`	Options for global mirroring
`bigpipe vip mirror`	Options for mirroring connection and persistence information on a virtual server.
`bigpipe snat mirror`	Options for mirroring secure NAT connections

Table 1.17 *Mirroring command in BIG/pipe*

Global mirroring on the BIG-IP Controller redundant system

You should enable mirroring on a redundant system at the global level before you can set mirroring of any specific types of connections or information. However, you can set specific types of mirroring and then enable global mirroring to begin mirroring. The syntax of the command for setting global mirroring is:

```
bigpipe mirror enable | disable | show
```

To enable mirroring on a redundant system, use the following command:

```
bigpipe mirror enable
```

To disable mirroring on a redundant system, use the following command:

```
bigpipe mirror disable
```

To show the current status of mirroring on a redundant system, use the following command:

```
bigpipe mirror show
```

Mirroring virtual server state

Mirroring provides seamless recovery for current connections, persistence information, SSL persistence, or sticky persistence when a BIG-IP Controller fails. When you use the mirroring feature, the standby controller maintains the same state information as the active controller. Transactions such as FTP file transfers continue as though uninterrupted.

Since mirroring is not intended to be used for all connections and persistence, it must be specifically enabled for each virtual server.

To control mirroring for a virtual server, use the **bigpipe vip mirror** command to enable or disable mirroring of persistence information, or connections, or both. The syntax of the command is:

```
bigpipe vip <virt addr>:<port> mirror [ persist | conn ] \
    enable | disable
```

Use **persist** to mirror persistence information for the virtual server. Use **conn** to mirror connection information for the virtual server. To display the current mirroring setting for a virtual server, use the following syntax:

```
bigpipe vip <virt addr>:<port> mirror [ persist | conn ] show
```

If you do not specify either **persist**, for persistent information, or **conn**, for connection information, the BIG-IP Controller assumes that you want to display both types of information.

Mirroring SNAT connections

SNAT connections are mirrored only if specifically enabled. You can enable SNAT connection mirroring by specific node address, and also by enabling mirroring on the default SNAT address. Use the following syntax to enable SNAT connection mirroring on a specific address:

```
bigpipe snat <node addr> [...<node addr>] mirror enable | disable
```

In the following example, the **enable** option turns on SNAT connection mirroring to the standby controller for SNAT connections originating from 192.168.225.100.

```
bigpipe snat 192.168.225.100 mirror enable
```

Use the following syntax to enable SNAT connection mirroring the default SNAT address:

```
bigpipe snat default mirror enable | disable
```

Using gateway fail-safe

Fail-safe features on the BIG-IP Controller provide network failure detection based on network traffic. Gateway fail-safe monitors traffic between the active controller and the gateway router, protecting the system from a loss of the internet connection by triggering a fail-over when the gateway is unreachable for a specified duration.

You can configure gateway fail-safe in the Configuration utility or in BIG/db. If you configure gateway fail-safe in BIG/db, you can toggle it on and off with **bigpipe** commands.

Adding a gateway fail-safe check

When you can set up a gateway fail-safe check using the Configuration utility, you need to provide the following information:

- Name or IP address of the router (only one gateway can be configured for fail-safe)
- Time interval (seconds) between pings sent to the router
- Time-out period (seconds) to wait for replies before proceeding with fail-over

To configure gateway fail-safe in the Configuration utility

1. In the navigation pane, click the BIG-IP Controller icon. The BIG-IP System Properties screen opens.
2. In the Gateway Fail-safe section of the screen, make the following entries:
   - Click the **Enabled** box.
   - In the **Router** box, type the IP address of the router you want to ping.

- In the **Ping (seconds)** box, type the interval, in seconds, you want the BIG-IP Controller to wait before it pings the router.
- In the **Timeout (seconds)** box, type the timeout value, in seconds. If the router does not respond to the ping within the number of seconds specified, the gateway is marked **down**.

3. Click the **Apply** button.

To configure gateway fail-safe in BIG/db

To enable gateway fail-safe in BIG/db, you need to change the settings of three specific BIG/db database keys using the **bigdba** utility. The keys set the following values:

❖ The IP address of the router

❖ The ping interval

❖ The timeout period

To set these keys, type this command to open the BIG/db database:

```
bigdba
```

To set the IP address of the router, type the following entry, where <**gateway IP**> is the IP address, or host name, of the router you want to ping:

```
Local.Bigip.GatewayPinger.Ipaddr=<gateway IP>
```

To set the ping interval, type the following entry, where <**seconds**> is the number of seconds you want the BIG-IP Controller to wait before pinging the router:

```
Local.Bigip.GatewayPinger.Pinginterval=<seconds>
```

To set the timeout, type the following entry, where <**seconds**> is the number of seconds you want the BIG-IP Controller to wait before marking the router **down**:

```
Local.Bigip.GatewayPinger.Timeout=<seconds>
```

To close **bigdba** and save your changes, type this command and press the Enter key:

```
quit
```

For more information about BIG/db and using **bigdba**, see *Supported BIG/db configuration keys*, on page 7-1.

◆ Note

*After you make these changes, you must restart **bigd** to activate the gateway pinger.*

Enabling gateway fail-safe

Gateway fail-safe monitoring can be toggled **on** or **off** from the command line using the **bigpipe gateway** command.

For example, arm the gateway fail-safe using the following command:

```
bigpipe gateway failsafe arm
```

To disarm fail-safe on the gateway, enter the following command:

```
bigpipe gateway failsafe disarm
```

To see the current fail-safe status for the gateway, enter the following command:

```
bigpipe gateway failsafe show
```

Gateway fail-safe messages

The destination for gateway fail-safe messages is set in the standard syslog configuration (**/etc/syslog.conf**), which directs these messages to the file **/var/log/bigd**. Each message is also written to the BIG-IP Controller console (**/dev/console**).

Using network-based fail-over

Network-based fail-over allows you to configure your redundant BIG-IP Controller to use the network to determine the status of the active controller. Network-based fail-over can be used in addition to, or instead of, hard-wired fail-over.

To configure network fail-over in the Configuration utility

1. In the navigation pane, click the BIG-IP Controller icon. The BIG-IP System Properties screen opens.
2. In the Redundant Configuration section of the screen, click the **Network Failover Enabled** box.
3. Click the **Apply** button.

To Configure network-based fail-over in BIG/db

To enable network-based fail-over, you need to change the settings of specific BIG/db database keys using the **bigdba** utility. To enable network-based fail-over, the **Common.Sys.Failover.Network** key must be set to one (**1**). To set this value to one, type this command to open the BIG/db database:

```
bigdba
```

At the **bigdba** prompt, type the following entry:

```
Common.Sys.Failover.Network=1
```

To close **bigdba** and save your changes, type this command and press the Enter key:

```
quit
```

Other keys are available to lengthen the delay to detect the fail-over condition on the standby controller, and to lengthen the heart beat interval from the active unit. To change the time required for the standby unit to notice a failure in the active unit, set the following value using the **bigdba** utility (the default is three seconds):

```
Common.Bigip.Cluster.StandbyTimeoutSec=<value>
```

To change the heart beat interval from the active BIG-IP Controller, change the following value using **bigdba** (the default is one second):

```
Common.Bigip.Cluster.ActiveKeepAliveSec=<value>
```

For more information about BIG/db and using **bigdba**, see *Supported BIG/db configuration keys*, on page 7-1.

Setting a specific BIG-IP Controller to be the preferred active unit

Setting a preferred active controller means overlaying the basic behavior of a BIG-IP Controller with a preference toward being active. A controller that is set as the active controller becomes active whenever the two controllers negotiate for active status.

To clarify how this differs from default behavior, contrast the basic behavior of a BIG-IP Controller in the following description. Each of the two BIG-IP Controllers in a redundant system has a built-in tendency to try to become the active controller. Each system attempts to become the active controller at boot time; if you boot two BIG-IP Controllers at the same time, the one that becomes the active controller is the one that boots up first. In a redundant configuration, if the BIG-IP Controllers are not configured with a preference for being the active or standby controller, either controller can become the active controller by becoming active first.

The active or standby preference for the BIG-IP Controller is defined by setting the appropriate startup parameters for **sod** (the switch over daemon) in **/etc/rc.local**. For more details on **sod** startup and functioning, see the ***BIG-IP Controller Reference Guide***, *System Utilities*.

The following example shows how to set the controller to standby:

```
echo " sod.";    /usr/sbin/sod -force_slave 2> /dev/null
```

A controller that prefers to be standby can still become the active controller if it does not detect an active controller.

This example shows how to set a controller to active:

```
echo " sod.";    /usr/sbin/sod -force_master 2> /dev/null
```

A controller that prefers to be active can still serve as the standby controller when it is on a live redundant system that already has an active controller. For example, if an active controller that preferred to be active failed over and was taken out of service for repair, it could then go back into service as the standby controller until the next time the redundant system needed an active controller, for example, at reboot.

Setting up active-active redundant controllers

You can use the active-active feature to simultaneously load balance traffic for different virtual addresses on redundant BIG-IP Controllers. Performance improves when both BIG-IP Controllers are in active service at the same time. In active-active mode, you configure virtual servers to be served by one of the two controllers. If one controller fails, the remaining BIG-IP Controller assumes the virtual servers of the failed machine. For this configuration to work, each controller has its own unit ID number. Each virtual server, NAT, or SNAT you create includes a unit number designation that determines which active controller handles its connections.

Note

If you do not want to use this feature, redundant BIG-IP Controllers operate in active/standby mode by default.

WARNING

MAC masquerading is not supported in active-active mode.

Configuring an active-active system

The default mode for BIG-IP Controller redundant systems is active/standby. You must take several steps in order to use active-active mode on the redundant BIG-IP Controller system. Details follow this brief list.

1. Configure an additional shared IP alias on the internal interface for each unit. You must have two shared aliases for the redundant system.
2. Set the routing configuration on the servers load balanced by the active-active BIG-IP Controller system.
3. Make sure the BIG/db key **Local.Bigip.Failover.UnitId** is 1 for one of the controllers, and 2 for the other.
4. Enable active-active mode by setting the BIG/db key **Common.Bigip.Failover.ActiveMode** to **1**.

5. Define the virtual servers, NATs, and/or SNATs to run on either unit 2 or on unit 1.
6. Update the fail-over daemon (**/sbin/sod**) with the configuration changes made in BIG/db.
7. Synchronize the configuration.
8. Transition from active/standby to active-active.

Note

We recommend making all of these configuration changes on one controller and then synchronizing the configuration.

Step 1: Configure an additional shared IP alias

When you configure a redundant system, you enter a shared IP alias. In active/standby mode, this shared IP alias runs on the active controller. You can determine if you already have a shared IP alias by running the **bigpipe interface** command. If you have one, it is probably configured as belonging to unit one.

In an active-active configuration, each BIG-IP Controller must have a shared IP alias on the internal, source processing, interface. This is the address to which the servers behind the BIG-IP Controller route traffic. Since you already have a shared IP alias for one controller, add a shared IP alias for the other controller by using the **bigpipe ipalias** command. For example:

```
bigpipe ipalias exp1 172.20.10.2 netmask
   255.255.0.0 unit 2
```

If you do not have a shared IP alias for unit 1, add one using this command. To view the IP aliases for the controller, type the **bigpipe interface** command on the command line.

If the BIG-IP Controller fails over, its shared IP address is assumed by the remaining unit and the servers continue routing through the same IP address.

You can configure additional shared IP aliases on an external, destination processing, interface of each BIG-IP Controller, as well. This makes it possible for routers to route to a virtual server using vip noarp mode.

To configure the additional shared IP alias in the Configuration utility

1. In the navigation pane, click **NICs**.
   The Network Interface Cards screen opens.
2. On the Network Interface Cards screen, click the name of the interface you want to configure.
   The Network Interface Card properties screen opens. You must choose an internal (source processing) interface.
3. In the Redundant Configuration section, check for a Unit 1 Alias and a Unit 2 Alias.
4. If one of the unit aliases is not present, type in an alias for the unit.
5. Click the **Apply** button.

Repeat this procedure on the other controller or use the Sync Configuration option in the toolbar of the BIG-IP System Properties page. Note that these settings should be identical on both controllers.

Step 2: Configuring servers for active-active

The active-active feature causes some restrictions on the servers behind the BIG-IP Controllers. The servers must be logically segregated to accept connections from one BIG-IP Controller or the other. To do this, set the default route to the BIG-IP Controller IP alias (see *Step 1: Configure an additional shared IP alias*) from which it accepts connections. In the case of a fail-over, the surviving BIG-IP Controller assumes the internal IP alias of the failed machine, providing each server a default route.

Step 3: Check the BIG-IP Controller unit number

Using the **bigdba** utility, check the value of the BIG/db key **Local.Bigip.Failover.UnitId**. This value should be 1 for one of the controllers, and 2 for the other.

Each BIG-IP Controller in an active-active configuration requires a unit number: either a 1 or a 2. The First-Time Boot utility allows a user to specify a unit number for each BIG-IP Controller. In an active-active configuration, specify the unit number when you configure virtual addresses, NATs, and SNATs.

◆ Note

You can only set this value directly in BIG/db. It cannot be set in the Configuration utility.

To check the BIG-IP Controller unit number in the Configuration utility

Follow this procedure on each BIG-IP Controller in a redundant system to check the BIG-IP Controller unit number with the Configuration utility:

1. Open the Configuration utility.
2. In the navigation pane, check the description next to the BIG-IP Controller icon.
   The status of the controller is **Active** and the unit number is either 1 or 2.

Step 4: Active-active BIG/db configuration parameters

To enable active-active, you must set the **Common.Bigip.Failover.ActiveMode** key to one (1). To set this value to one, follow these steps:

Type the following command to open the BIG/db database:

```
bigdba
```

At the **bigdba** prompt, type the following entry:

```
Common.Bigip.Failover.ActiveMode=1
```

Type **quit** to exit BIG/db and save the configuration.

The default for this entry is **off** and fail-over runs in active/standby mode.

To enable active-active in the Configuration utility

Perform this procedure on the active controller first. After the active box is enabled, follow this procedure on the standby controller. After you perform this feature on the standby controller, wait 30 seconds and click the **Refresh** button (Microsoft Internet Explorer) or **Reload** button (Netscape Navigator) on the browser for both controllers.

1. In the navigation pane, click the BIG-IP Controller icon.
2. The BIG-IP Controller System Properties screen opens.
3. Click the **Active-Active Mode Enabled** check box.
4. Click the **Apply** button.

Step 5: Virtual address configuration

Both BIG-IP Controllers must have the exact same configuration file (**/etc/bigip.conf**). When a virtual server is defined, it must be defined with a unit number that specifies which BIG-IP Controller handles connections for the virtual server. Each BIG-IP Controller has a unit number, 1 or 2, and serves the virtual servers with corresponding unit numbers. If one of the BIG-IP Controllers fails over, the remaining BIG-IP Controller processes the connections for virtual servers for both units.

Defining virtual servers, NATs, and SNATs on active-active controllers

Use the following commands to define virtual servers, NATs, and SNATs on active-active controllers:

```
bigpipe vip <virt addr>:<port> define [unit <1|2>]
   <node addr>:<port>
bigpipe nat <internal_ip> to <external_ip> ... [unit <1|2>]
bigpipe snat map <orig_ip> to <trans_ip> ... [unit <1|2>]
```

◆ Note

If not specified, the unit number defaults to 1.

Each BIG-IP Controller in an active-active configuration requires a unit number: either a 1 or a 2. Use the First-Time Boot utility to specify a unit number for each BIG-IP Controller. If you do not specify a unit number, the unit number for the virtual server defaults to 1.

◆ Note

You must specify the unit number when defining virtual servers, NATs, and SNATs. You cannot add the unit number at a later time without redefining the virtual server, NAT, or SNAT.

To define virtual servers, NATs, and SNATs on active-active controllers in the Configuration utility

The following example illustrates the unit ID number in a virtual server definition. Although the steps to create a NAT or SNAT are slightly different, the unit ID number serves the same purpose.

1. In the navigation pane, click **Virtual Servers**.
   The Virtual Servers screen opens.
2. In the toolbar, click the **Add Virtual Server** button.
3. Type in the address, netmask, and port for the virtual server.
4. Click the **Unit ID** list. Select the unit number for the virtual server.
5. The connections served by this virtual server are managed by the controller assigned this unit ID.
6. Complete the Resources section of the screen. For more information about individual settings, refer to the online help.
7. Click the **Apply** button.

Step 6: Update the fail-over daemon (/sbin/sod) with the configuration changes made in BIG/db

Active-active mode is implemented by the fail-over daemon (**/sbin/sod**). If you change a BIG/db key that affects the fail-over daemon (keys that contain the word Failover) the fail-over daemon needs to be updated with the change. To update the fail-over daemon, type the following command:

```
bigpipe failover init
```

Step 7: Synchronize the configuration

After you complete steps 1 through 6 on each controller in the active-active system, synchronize the configurations on the controllers with the Configuration utility, or from the command line.

To synchronize the configuration in the Configuration utility

1. In the navigation pane, click the BIG-IP Controller icon. The BIG-IP Properties screen opens.
2. In the toolbar, click the **Sync Configuration** button. The Synchronize Configuration screen opens.
3. Click the **Synchronize** button.

To synchronize the configuration from the command line

To synchronize the configuration between two controllers from the command line, use the following command:

```
bigpipe configsync all
```

Step 8: Transition from active/standby to active-active

To transition from active/standby to active-active, type the following command on the active BIG-IP Controller:

```
bigpipe failover standby
```

This command puts the active BIG-IP Controller into partial active-active mode. To complete the transition, type in the following command on the other BIG-IP Controller which now considers itself the active unit.

```
bigpipe failover standby
```

Now both units are in active-active mode.

◆ Note

This step is not required if you enable active-active in the Configuration utility. The transition is made during Step 4: Active-active BIG/db configuration parameters, on page 1-104.

Active-active system fail-over

Before a failure in an active-active installation, one BIG-IP Controller is servicing all requests for virtual servers configured on unit 1, and the other BIG-IP Controller is servicing all requests for virtual servers configured on unit 2. If one of the BIG-IP Controllers fails, the remaining BIG-IP Controller handles all requests for virtual servers configured to run on unit 1 and also those configured to run on unit 2. In other words, the surviving BIG-IP Controller is acting as both units 1 and 2.

If the BIG-IP Controller that failed reboots, it re-assumes connections for the unit number with which it was configured. The BIG-IP Controller that was running as both units stops accepting connections for the unit number that has resumed service. Both machines are now active.

When the unit that was running both unit numbers surrenders a unit number to the rebooted machine, all connections are lost that are now supposed to run on the rebooted machine, unless they were mirrored connections.

Disabling automatic fail back

In some cases, you may not want connections to automatically fail back. The fact that a machine has resumed operation may not be reason enough to disrupt connections that are running on the BIG-

IP Controller serving as both units. Note that because of addressing issues, it is not possible to slowly drain away connections from the machine that was running as both units, giving new requests to the recently rebooted machine.

To disable automatic fail back, set the BIG/db key **Common.Bigip.Failover.ManFailBack** to 1. When you set this key to **1**, a BIG-IP Controller running as both units does not surrender a unit number to a rebooted peer until it receives the **bigpipe failover failback** command. By default, this key is not set.

Taking an active-active controller out of service

You can use the bigpipe failover standby command to place an active controller in standby mode. In active-active mode, type the following command to place a one of the active controllers in standby mode:

```
bigpipe failover standby
```

This command causes the BIG-IP Controller to surrender its unit number to its peer. That is, its peer now becomes both units 1 and 2, the BIG-IP Controller appears out of service from a fail-over perspective, it has no unit numbers. You can make any changes, such as configuration changes, before causing the machine to resume normal operation.

Placing an active-active controller back in service if automatic failback is disabled

If the **Common.Bigip.Failover.ManFailBack** key is set to **0** (off), normal operation is restored when you issue a **bigpipe failover failback** command on the controller with no unit number.

In active-active mode, type the following command to place a standby controller back in service:

```
bigpipe failover failback
```

This command causes the BIG-IP Controller to resume its unit number. That is, the peer now relinquishes the unit number of the controller that has resumed service.

However, if the **Common.Bigip.Failover.ManFailBack** key is set to **1** (on), normal operations are restored when you issue a **bigpipe failover failback** command on the controller running with both unit numbers.

Additional active-active BIG/db configuration parameters

There are several new BIG/db parameters for active-active mode.

Common.Bigip.Failover.ActiveMode
Set this BIG/db parameter to **1** to enable active-active mode. The default setting is off, and redundant systems run in active/standby mode.

Local.Bigip.Failover.UnitId
This is the default unit number of the BIG-IP Controller. This value is set by the First-Time Boot utility or when you upgrade your controllers to this version of the BIG-IP Controller.

Common.Bigip.Failover.ManFailBack
This is set to **1** so that manual intervention is required (the **bigpipe failover failback** command is issued) before a BIG-IP Controller running both unit numbers surrenders a unit number to its peer. This feature is **off** by default, fail-back is automatic. For more details, see the section *Active-active system fail-over*, on page 1-108.

Common.Bigip.Failover.NoSyncTime
Set this to **1** if you do not want to synchronize the time of the two BIG-IP Controllers. Normally, their time is synchronized. For some cases, this is not desirable, for example, if you are running **ntpd**.

Common.Bigip.Failover.AwaitPeerDeadDelay
The BIG-IP Controller checks to see that its peer is still alive at this rate (in seconds). The default value for this parameter is one second.

Common.Bigip.Failover.AwaitPeerAliveDelay
Check status of a peer BIG-IP Controller while waiting for it to come to life with this frequency (in seconds). The default value of this parameter is three seconds.

Common.Bigip.Failover.DbgFile
If a file name is specified, the fail-over daemon logs state change information in this file. This value is not set by default.

Common.Bigip.Failover.PrintPeerState
Causes the fail-over daemon to periodically write the state of its connections to its peer (hard-wired and/or network) to the log file **Common.Bigip.Failover.DbgFile**.

Additional commands for displaying active vs. mirrored data

The **dump** commands explicitly show those connections (and other objects) that are active on the BIG-IP Controller, and those that are standby connections for the peer BIG-IP Controller. In prior versions of the BIG-IP Controller, one controller is the active unit and the other is the standby. When the **bigpipe conn dump** command is issued on the active unit, each of the connections shown is active. Similarly, when the **bigpipe conn dump** command is issued on the standby unit, it is clear that each of the connections listed is a standby connection. These standby connections are created by mirroring the active connections on the standby unit.

In an active-active installation, each unit can be considered a standby for its peer BIG-IP Controller. By default, the **dump** command only shows items that are active on the given unit. To see standby items you must use the **mirror** qualifier. You can use the following commands with the mirror option:

```
bigpipe conn dump [mirror]
bigpipe vip persist dump [mirror]
bigpipe sticky dump [mirror]
```

Also, the **bigpipe snat show** command output has been modified to show whether a connection listed is an active connection or a mirror connection.

Specific active-active bigpipe commands

Several specific commands are included in **bigpipe** to reflect new or changed functionality.

`bigpipe failover init`

This command causes the fail-over daemon (**/sbin/sod**) to read the BIG/db database and refresh its parameters.

bigpipe failover failback

After a **bigpipe failover standby** command is issued, issue this command to allow the BIG-IP Controller to resume normal operation. If manual fail back is enabled, this command causes a BIG-IP Controller that is running as both units to release a unit number to its peer unit when the peer becomes active. You can use the following commands to view the unit number on the controller you are logged into:

`bigpipe unit [show]`

To view the unit number, or numbers, of the peer BIG-IP Controllers in a redundant system, type the following command:

`bigpipe unit peer [show]`

Running mixed versions of BIG-IP Controller software in active-active mode

The BIG-IP Controller provides the option to install a new version of the BIG-IP Controller software on one BIG-IP Controller, while the other BIG-IP Controller runs a previous production version of the software. This allows you to fail back and forth between the two units, testing the new software yet having the ability to return to the prior installation.

This is possible with the new fail-over software in version 3.1, whether using active-active mode or active/standby mode. However, there are some exceptions:

State mirroring is not compatible between the version 3.0 and prior versions of the software. Network fail-over is also not compatible.

If you are running the BIG-IP Controller version 3.1 in active-active mode, you should assign unit two to the BIG-IP Controller running version 3.1.

Returning an active-active installation to active/standby mode

Returning to active/standby mode from active-active mode is relatively simple in that only a few things need be undone.

1. Enable active/standby mode by setting the BIG/db key **Common.Bigip.Failover.ActiveMode** to **0**.
2. Update the fail-over daemon with the change by typing **bigpipe failover init**.
3. To synchronize the configuration, type the command **bigpipe configsync all**.
4. Since each BIG-IP Controller is an active unit, type the command **bigpipe failover standby** on each controller. This transitions each controller into active/standby mode.

When in active/standby mode, the active BIG-IP Controller runs all objects (virtual servers, SNATs and NATs) that are defined to run on unit 1 or unit 2. It is not necessary to redefine virtual servers, SNATS, or NATs when you transition from active-active mode to active/standby mode.

Rule

You can create a rule that references two or more load balancing pools. In other words, a rule selects a pool for a virtual server. A rule is referenced by 1- to 31-character name. When a packet arrives that is destined for a virtual server that does not match a current connection, the BIG-IP Controller can select a pool by evaluating a virtual server rule to pick a node pool. The rule is configured to ask true or false questions such as:

- HTTP header load-balancing: Does the packet data contain an HTTP request with a URI ending in cgi?
- IP header load balancing: Does the source address of the packet begin with the octet 206?

The attributes you can configure for a rule are in Table 1.18

Attributes	Description
Pool selection based on HTTP request data	This type of rule sends connections to a pool, or pools based on HTTP header information you specify.
Pool selection based on IP packet header information	This type of rule sends connections to a pool, or pools based on IP header information you specify.
Cache control rule	This type of rule is any rule that contains a cache statement. A cache control rule selects a pool based on HTTP header data. You cannot use it with FTP.

Table 1.18 *The attributes you can configure for a rule.*

Pool selection based on HTTP request data

The rule specifies what action the BIG-IP Controller takes depending on whether a question is answered true or false. The rule may either select a pool or ask another question. For example, you may want a rule that states if the packet data contains an HTTP request with a URI ending in **cgi**, then load balance using the pool **cgi_pool**. Otherwise, load balance using the pool **default_pool**.

Figure 1.7 shows a rule with an HTTP request variable that illustrates this example:

```
rule cgi_rule {
    if (http_uri ends_with "cgi") {
       use ( cgi_pool )
    }
    else {
       use ( default_pool )
    }
}
```

Figure 1.7 *A rule based on an HTTP header variable*

Load balancing normally happens right after the BIG-IP Controller receives a packet that does not match a current connection. However, in the case of an HTTP request, the first packet is a TCP SYN packet that does not contain the HTTP request. In this case, the BIG-IP Controller proxies the TCP handshake with the client and begins evaluating the rule again when the packet containing the HTTP request is received. When a pool has been selected and a server node selected, the BIG-IP Controller proxies the TCP handshake with the server node and then passes traffic normally.

Pool selection based on IP packet header information

In addition to the HTTP variables, you can also use IP packet header information such as the **client_addr** or **ip_protocol** variables to select a pool. For example, if you want to load balance based on part of the client's IP address, you may want a rule that states:

"All client requests with the first byte of their source address equal to **206** will load balance using a pool named **clients_from_206** pool. All other requests will load balance using a pool named **other_clients_pool**."

Figure 1.8 shows a rule based on the client IP address variable that illustrates this example:

```
rule clients_from_206_rule {
    if ( client_addr equals 206.0.0.0 netmask 255.0.0.0 ) {
       use ( clients_from_206 )
    }
    else {
       use ( other_clients_pool )
    }
}
```

Figure 1.8 *A rule based on the client address variable*

Statements

A rule consists of statements. Rules support three kinds of statements:

- An **if** statement asks a true or false question and, depending on the answer, decides what to do next
- A **discard** statement discards the request. This statement must be conditionally associated with an **if** statement
- A **use** statement uses a selected pool for load balancing. This statement must be conditionally associated with an **if** statement
- A **cache** statement uses a selected pool for load balancing. This statement can be conditionally associated with an **if** statement

The three possible statements expressed in command line syntax are:

```
if (<question>) {<statement>} [else {<statement>}]
discard
use ( <pool_name> )
cache ( <expressions> )
```

Questions (expressions)

A question or expression is asked by an **if** statement and has a true or false answer. A question or expression has two parts: a predicate (**operator**), and one or two subjects (**operands**).

There are two types of subjects (operands); some subjects change and some subjects stay the same.

- Changing subjects are called **variable operands**.
- Subjects that stay the same are called **constant operands**.

A question, or ***expression***, asks questions about variable operands by comparing their current value to constant operands with relational operators.

Constant operands

Possible constant operands are:

- IP protocol constants, for example:
  UDP or **TCP**
- IP addresses expressed in masked dot notation, for example:
  206.0.0.0 netmask 255.0.0.0
- Strings of ASCII characters, for example:
  "pictures/bigip.gif"
- Regular expression strings

Variable operands (variables)

Since variable operands change their value, they need to be referred to by a constant descriptive name. The variables available depend on the context in which the rule containing them is evaluated. Possible variable operands are:

- IP packet header variables, such as:
  - Client request source IP address with the **client_addr** variable. The **client_addr** variable is replaced with an unmasked IP address.
  - IP protocol, UDP or TCP, with the **ip_protocol** variable. The **ip_protocol** variable is replaced with either the **UDP** or **TCP** protocol value.

- HTTP request strings (see *HTTP request string variables*, on page 1-119). All HTTP request string variables are replaced with string literals.

The evaluation of a rule is triggered by the arrival of a packet. Therefore, variables in the rule may refer to features of the triggering packet. In the case of a rule containing questions about an HTTP request, the rule is evaluated in the context of the triggering TCP SYN packet until the first HTTP request question is encountered. After the proxy, the rule continues evaluation in the context of the HTTP request packet, and variables may refer to this packet. Before a variable is compared to the constant in a relational expression, it is replaced with its current value.

In a rule, relational operators compare two operands to form relational expressions. Possible relational operators and expressions are described in Table 1.19:

Expression	Relational Operator
Are two IP addresses equal?	`<address> equals <address>`
Do a string and a regular expression match?	`<variable_operand> matches_regex <regular_expression>`
Are two strings identical?	`<string> equals <string>`
Is the second string a suffix of the first string?	`<variable_operand> ends_with <string>`
Is the second string a prefix of the first string?	`<variable_operand> starts_with <string>`
Does the first string contain the second string?	`<variable_operand> contains <literal_string>`

Table 1.19 *The relational operators*

In a rule, logical operators modify an expression or connect two expressions together to form a logical expression. Possible logical operators and expressions are described in Table 1.20:

Expression	Logical Operator
Is the expression not true?	`not <expression>`
Are both expressions true?	`<expression> and <expression>`
Is either expression true?	`<expression> or <expression>`

Table 1.20 *The logical operators*

HTTP request string variables

HTTP request variables are referred to in command line syntax by a predefined set of names. Internally, an HTTP request variable points to a method for extracting the desired string from the current HTTP request header data. Before an HTTP request variable is used in a relational expression, it is replaced with the extracted string. The allowed variable names are:

http_method
The **http_method** is the action of the HTTP request. Common values are **GET** or **POST**.

http_uri
The **http_uri** is the URL, but does not include the protocol and the fully qualified domain name (FQDN). For example, if the URL is "http://www.url.com/buy.asp", then the URI is "/buy.asp".

http_version
The **http_version** is the HTTP protocol version string. Possible values are **HTTP/1.0** or **HTTP/1.1**.

http_host
The **http_host** is the value in the **Host:** header of the HTTP request. It indicates the actual FQDN that the client requested. Possible values are a FQDN or a host IP address in dot notation.

http_cookie <cookie name>
The HTTP cookie header is value in the **Cookie:** for the specified cookie name. An HTTP cookie header line can contain one or more cookie name value pairs. The **http_cookie <cookie name>** variable evaluates to the value of the cookie with the name **<cookie name>**.

For example, given a request with the following cookie header line:

```
Cookie: green-cookie=4; blue-cookie=horses
```

The variable **http_cookie blue-cookie** evaluates to the string **horses**. The variable **http_cookie green-cookie** evaluates to the string **4**.

http_header <header_tag_string>
The variable **http_header** evaluates the string following an HTTP header tag that you specify. For example, you can specify the **http_host** variable with the **http_header** variable. In a rule specification, if you wanted to load balance based on the host name "andrew" it might look like this:

```
if ( http_header "Host" starts_with "andrew" )  { use ( andrew_pool
   ) } else { use ( main_pool ) }
```

Configuring rules

You can create rules from the command line or with the Configuration utility. Each of these methods is described in this section.

To add a rule in the Configuration utility

1. In the navigation pane, click **Rules**.
   This opens the Rules screen.
2. In the toolbar, click the **Add Rule** button.
   The Add Rule screen opens.
3. In the **Rule Name** box, type in the name you want to use for the rule.
4. In the **Text** box, type in a rule. Note that you should not enclose the rule with curly braces { } as you do when you create a rule directly in the **bigip.conf** file.

5. You can type in the rule as an unbroken line, or you can use the Enter key to add line breaks.
6. Click the **Add** button to add the rule to the BIG-IP Controller configuration.

To define a rule from the command line

To define a rule from the command line, use the following syntax:

```
bigpipe rule <rule_name> ' { <if statement> } '
```

For more information about the elements of a rule, see Table 1.22, on page 1-122.

Configuring virtual servers that reference rules

Using either the Configuration utility or the command line, you can define a virtual server that references a rule.

To configure a virtual server that references a rule in the Configuration utility

1. In the navigation pane, click **Virtual Servers**.
   The Virtual Servers screen opens.
2. Add the attributes you want for the virtual server such as Address, Port, Unit ID, and Interface.
3. In the Resources section, click **Rule**.
4. In the Rule list, select the rule you want to apply to the virtual server.
5. Click the **Apply** button.

To configure a virtual server that references a rule from the command line

There are several elements required for defining a virtual server that references a rule from the command line:

```
bigpipe vip <virt_serv_key> { <vip_options> <rule_name_reference> }
```

Each of these elements is described in Table 1.21:

Rule element	Description
<virt_serv_key>	A virtual server key definition: `<virtual_address>:<virt_port> [<interface_name>] [unit <ID>]`
<vip_options>	Virtual server options such as IP netmask and broadcast address. For more information, see the ***BIG-IP Controller Reference Guide***, *bigpipe Command Reference*.
<rule_name_reference>	A rule name reference. Rule names are strings of 1 to 31 characters. `use rule <rule_name>`

Table 1.21 *The command line rule elements*

Note

You must define a pool before you can define a rule that references the pool.

Table 1.22 contains descriptions of all the elements you can use to create rules.

Element	Description
A rule definition is	`rule { <if_statement> }`

A statement is	`<use_statement>`
	`<if_statement>`
	`discard`
	`<cache_statement>`

A **use** statement	`use ( <pool_name> )`

An **if** statement	`if ( <expression> ) { <statement> }`
	`[ { else <statement> } ]`
	`<cache_statement>`

An expression	`<literal>`
	`<variable>`
	`( <expression> )`
	`exist <variable>`
	`not <expression>`
	`<expression> <binary_operator> <expression>`

Table 1.22 *The elements you can use to construct rules*

Element	Description
IP protocol constants	**UDP** **TCP**
literal	**<regex_literal>** **<string_literal>** **<address_literal>**
A regular expression literal	Is a string of 1 to 63 characters enclosed in quotes that may contain regular expressions
A string literal	Is a string of 1 to 63 characters enclosed in quotes
An address literal	**<dot_notation_longword> [netmask** **<dot_notation_longword>]**
Dot notation longword	**<0-255>.<0-255>.<0-255>.<0-255>**
variable	**http_method** **http_header <header tag>** **http_version** **http_uri** **http_host** **http_cookie <cookie_name>** **client_addr** **ip_protocol**
binary operator	**or** **and** **contains** **matches** **equals** **starts_with** **ends_with** **matches_regex**
Element	**Description**

Table 1.22 *The elements you can use to construct rules*

Cache statement syntax

A cache statement may be either the only statement in a rule or it may be nested within an **if** statement. The syntax of a cache statement is:

```
cache ( <expression> ) {
    origin_pool <pool_name>
    cache_pool <pool_name>
    [ hot_pool <pool_name> ]
    [ hot_threshold <hit_rate> ]
    [ cool_threshold <hit_rate> ]
    [ hit_period <seconds> ]
    [ content_hash_size <sets_in_content_hash> ]
 }
```

Figure 1.9 *An example of cache statement syntax*

The following table describes the cache rule syntax:

Rule Syntax	Description
origin_pool <pool_name>	This required attribute specifies a pool of servers with all the content to which requests are load balanced when the requested content is not cacheable or when all the cache servers are unavailable or when you use a BIG-IP Controller to redirect a miss request from a cache.
cache_pool <pool_name>	This required attribute specifies a pool of cache servers to which requests are directed to optimize cache performance.
hot_pool <pool_name>	This optional attribute specifies a pool of servers that contain content to which requests are load balanced when the requested content is frequently requested (hot). If you specify any of following attributes in this table, the **hot_pool** attribute is required.
hot_threshold <hit_rate>	This optional attribute specifies the minimum number of requests for content that cause the content to change from cool to hot at the end of the period (**hit_period**).

Table 1.23 *Description of rule syntax*

Rule Syntax	Description
cool_threshold <hit_rate>	This optional attribute specifies the maximum number of requests for specified content that cause the content to change from hot to cool at the end of the period.
hit_period <seconds>	This optional attribute specifies the period in seconds over which to count requests for particular content before deciding whether to change the hot or cool state of the content.
content_hash_size <sets_in_content_hash>	This optional attribute specifies the number subsets into which the content is divided when calculating whether content is hot or cool. The requests for all content in the same subset are summed and a single hot or cool state is assigned to each subset. This attribute should be within the same order of magnitude as the actual number of requests possible. For example, if the entire site is composed of 500,000 pieces of content, a **content_hash_size** of 100,000 would be typical.

Table 1.23 *Description of rule syntax*

A cache statement returns either the origin pool, the hot pool, or the cache pool. When the cache pool is selected, it is accompanied by the indicated node address and port. When a rule returns both a pool and a node, the BIG-IP Controller does not do any additional load balancing or persistence processing.

The following is an example of a rule containing a cache rule statement.

```
rule my_rule {
    if ( http_host starts_with "dogfood" ) {
        cache ( http_uri ends_with "html" or http_uri ends_with
"gif" ) {
        origin_pool origin_server
        cache_pool cache_servers
        hot_pool cache_servers
        hot_threshold 100
        cool_threshold 10
        hit_period 60
        content_hash_size 1024
        }
    }
    else {
        use ( catfood_servers )
    }
}
```

Figure 1.10 *An example of a cache load balancing rule*

Using a default SNAT if your origin server is external to the BIG-IP Controller

If your origin server is external to the BIG-IP Controller, you must configure a default SNAT, enable source translation on the external interface, and set each origin server node address to **remote** in addition to creating a cache rule. This configuration allows source translation of non-cachable requests directed to the origin server without causing other incoming traffic to be source translated. To create a default SNAT, type the following command:

```
bigpipe snat map default to <trans_ip_addr>
```

Substitute the IP address of the default SNAT for **<trans_ip_addr>**.

To enable source translation on the external interface

To enable source translation on the external interface, type the following command:

```
bigpipe interface <ext_interface> source_translation enable
```

Substitute the name of the external interface for **<ext_interface>**.

To set the remote attribute for each origin server in the BIG-IP Controller configuration

To add the remote origin server to the BIG-IP Controller configuration, use the following command:

```
bigpipe node <origin_ip> remote
```

Substitute the IP address of the origin server for **<origin_ip>**.

Additional rule examples

This section includes additional examples or rules. The following rule examples are included:

- Cookie rule
- Language rule
- Cacheable contents rule
- AOL rule
- Protocol specific rule

Cookie rule

This example is a cookie rule that load balances based on the user ID that contains the word **VIP**.

```
if ( exists http_cookie "user-id" and
     http_cookie "user-id" contains "VIP" ) {
   use ( vip_pool )
}
else {
   use ( other_pool )
}
```

Figure 1.11 *An example cookie rule*

Language rule

This is an example of a rule that load balances based on the language requested by the browser:

```
if ( exists http_header "Accept-Language" ) {
   if ( http_header "Accept-Language" equals "fr" ) {
      use ( french_pool )
   }
   else {
      if ( http_header "Accept-Language" equals "sp" ) {
         use (spanish_pool )
      }
      else {
         use ( english_pool )
      }
   }
else {
   use ( english_pool )
}
```

Figure 1.12 *An example of a rule that load balances based on the language requested by the browser*

Cache content rule

This is an example of a rule that you can use to send cache content, such as gifs, to a specific pool.

```
if ( http_uri ends_with "gif" or
     http_uri ends_with "html" ) {
   use ( cache_pool )
}
else {
   use ( server_pool )
}
```

Figure 1.13 *An example of a cache content rule*

AOL rule

This is an example of a rule that you can use to load balance incoming AOL connections.

```
port 80 443 enable

pool aol_pool {
   lb_method priority_member
   member 12.0.0.31:80 priority 4
   member 12.0.0.32:80 priority 3
   member 12.0.0.33:80 priority 2
   member 12.0.0.3:80 priority 1
}
pool other_pool {
   lb_method round_robin
   member 12.0.0.31:80
   member 12.0.0.32:80
   member 12.0.0.33:80
   member 12.0.0.3:80
}
pool aol_pool_https {
   lb_method priority_member
   member 12.0.0.31:443 priority 4
   member 12.0.0.32:443 priority 3
   member 12.0.0.33:443 priority 2
   member 12.0.0.3:443 priority 1
}
pool other_pool_https{
   lb_method round_robin
   member 12.0.0.31:443
   member 12.0.0.32:443
   member 12.0.0.33:443
   member 12.0.0.3:443
}
rule aol_rule {
   if (    client_addr equals 152.163.128.0 netmask 255.255.128.0
        or client_addr equals 195.93.0.0    netmask 255.255.254.0
        or client_addr equals 205.188.128.0 netmask 255.255.128.0 ) {
      use ( aol_pool )
   }
   else {
      use ( other_pool)
   }
}
rule aol_rule_https {
   if (    client_addr equals 152.163.128.0 netmask 255.255.128.0
        or client_addr equals 195.93.0.0    netmask 255.255.254.0
        or client_addr equals 205.188.128.0 netmask 255.255.128.0 ) {
      use ( aol_pool_https )
   }
   else {
      use ( other_pool_https)
   }
}
vip 15.0.140.1:80  { use rule aol_rule }
vip 15.0.140.1:443 { use rule aol_rule_https special ssl 30 }
```

Figure 1.14 *An example of an AOL rule*

IP protocol specific rule

This is an example of a rule that you can use to send TCP DNS to the pool **tcp_pool** and UDP DNS to the pool **udp_pool**.

```
rule myrule {
   if ( ip_protocol equals UDP ) {
     use ( udp_pool )
   }
   else {
      use ( tcp_pool )
   }
```

Figure 1.15 *An example of an IP protocol rule*

Comparing load balancing configurations

You can use the method from previous versions of the BIG-IP Controller to define a virtual server with a single node list. However, with this version of the BIG-IP Controller, node list virtual servers are being phased out. Node lists use the global load balancing mode set on the BIG-IP Controller. The global mode cannot be set to the **ratio_member**, **priority_member**, **least_conn_member**, **observed_member**, or **predictive_member** load balancing modes. For an example of a node list, see Figure 1.16.

```
lb ratio
vip 15.0.140.1:80 {
    define 12.0.0.44:80 12.0.0.45:80
}
ratio {
    12.0.0.44
} 1
ratio {
    12.0.0.45
} 2
```

Figure 1.16 *The node list method of defining virtual servers*

In contrast to a node list virtual server, you can share pools with a number of virtual servers on the BIG-IP Controller. For example, Figure 1.17 shows the **gif_pool** shared by two virtual servers:

```
pool cgi_pool {
    lb_method ratio_member
    member 12.0.0.44:80 ratio 1
    member 12.0.0.45:80 ratio 2
}
pool gif_pool {
    lb_method ratio_member
    member 12.0.0.44:80 ratio 1
    member 12.0.0.45:80 ratio 3
}
rule http_rule {
    if ( http_uri ends_with "gif" ) {
        use ( gif_pool )
    }
    else {
        use ( cgi_pool )
    }
}

vip 15.0.140.1:80 {
    netmask 255.255.0.0 broadcast 15.0.255.255
    use rule http_rule
}
vip 15.0.140.2:80 {
    netmask 255.255.0.0 broadcast 15.0.255.255
    use pool gif_pool
}
```

Figure 1.17 *An example of a pool shared by two virtual servers*

SNAT

When you define secure network address translations (SNATs), you can assign a single SNAT address to multiple nodes. Note that a SNAT address does not necessarily have to be unique; for example, it can match the IP address of a virtual server.

SNAT addresses have global properties that apply to all SNATs that you define in the BIG-IP Controller configuration as well as to the SNAT mappings you define. You can configure SNATs in the Configuration utility or from the command line.

The attributes you can configure for a SNAT are in Table 1.24.

Attributes	Description
Global SNAT properties	Before you can configure a SNAT, you must configure global properties for all SNATs on the BIG-IP Controller.
Default SNAT	If you do not wish to configure specific SNATs, you can configure a default SNAT.
Individual SNAT	You can configure individual SNATs for specific hosts in the network.

Table 1.24 *The attributes you can configure for a SNAT*

Setting SNAT global properties

The SNAT feature supports three global properties that apply to all SNAT addresses:

- **Connection limits**
  The connection limit applies to each node that uses a SNAT, and each individual SNAT can have a maximum of 50,000 simultaneous connections.
- **TCP idle connection timeout**
  This timer defines the number of seconds that TCP connections initiated using a SNAT address are allowed to remain idle before being automatically disconnected.

❖ **UDP idle connection timeout**
This timer defines the number of seconds that UDP connections initiated using a SNAT address are allowed to remain idle before being automatically disconnected. This value should not be set to **0**.

To configure SNAT global properties in the Configuration utility

1. In the navigation pane, click **Secure NATs**.
   The Secure Network Address Translations screen opens.
2. In the **Connection Limit** box, type the maximum number of connections you want to allow to each node using a SNAT. To turn connection limits off, set the limit to **0**. If you turn connection limits on, keep in mind that each SNAT can support only 50,000 simultaneous connections.
3. In the **TCP Idle Connections** box, type the number of seconds that TCP connections initiated by a node using a SNAT are allowed to remain idle.
4. In the **UDP Idle Connections** box, type the number of seconds that UDP connections initiated by a node using a SNAT are allowed to remain idle. This value should not be set to **0**.
5. Click the **Apply** button.

To configure SNAT global properties on the command line

Configuring global properties for a SNAT requires that you enter three **bigpipe** commands. The following command sets the maximum number of connections you want to allow for each node using a SNAT.

```
bigpipe snat limit <value>
```

The following commands set the TCP and UDP idle connection timeouts:

```
bigpipe snat timeout tcp <seconds>
bigpipe snat timeout udp <seconds>
```

Configuring SNAT address mappings

Once you have configured the SNAT global properties, you can configure SNAT address mappings. The SNAT address mappings define each SNAT address, and also define the node or group of nodes that uses the SNAT address. Note that a SNAT address does not necessarily have to be unique; for example, it can match the IP address of a virtual server. A SNAT address cannot match an address already in use by a NAT, SNAT, or BIG-IP Controller address.

To configure a SNAT mapping in the Configuration utility

1. In the navigation pane, click **Secure NATs**.
   The Secure Network Address Translations screen opens.
2. On the toolbar, click **Add SNAT**.
   The Add SNAT screen opens.
3. In the **Translation Address** box, type the IP address that you want to use as the alias IP address for the node(s).
4. In the **Interface** box, you can select the external interface (destination processing) on which the SNAT address is to be used. Note that this setting applies only if your BIG-IP Controller has more than one destination processing interface.
5. In the **Original Address** box, type the IP address of the node or nodes that are assigned to the SNAT. Click the add button (>>) to add the address to the **Current List**.
6. To remove an address from the **Current List**, click the remove button (<<).
7. Click the **Apply** button.

To configure a SNAT mapping on the command line

The **bigpipe snat** command defines one SNAT for one or more node addresses.

```
bigpipe snat map <node addr>... <node addr> to <SNAT_addr>
```

For example, the command below defines a secure network address translation for two nodes:

```
bigpipe snat map 192.168.75.50 192.168.75.51 to 192.168.100.10
```

Defining the default SNAT

Use the following syntax to define the default SNAT. If you use the netmask parameter and it is different from the external interface default netmask, the command sets the netmask and derives the broadcast address.

You can use the **unit** <**unit ID**> parameter to specify a unit in an active-active redundant configuration.

```
bigpipe snat map default to <translated_ip> [<ifname>] [unit <unit
   ID>] [netmask <ip>]
```

Creating individual SNAT addresses

Use the following command syntax to create a SNAT mapping:

```
bigpipe snat map <orig_ip> [...<orig_ip>] to \
   <SNAT ip> [<ifname>] [unit <unit ID>] [netmask <ip>]
```

If the netmask is different from the external interface default netmask, the command sets the netmask and derives the broadcast address.

Deleting SNAT Addresses

The following syntax deletes a specific SNAT:

```
bigpipe snat <SNAT ip> | default delete
```

Showing SNAT mappings

The following **bigpipe** command shows mappings:

```
bigpipe snat [<SNAT ip>] [...<SNAT ip>] show
bigpipe snat default show
```

The **<SNAT ip>** can be either the translated or original IP address of the SNAT.

The following command shows the current SNAT connections:

```
bigpipe snat [<SNAT ip>] [...<SNAT ip>] dump [ verbose ]
```

```
bigpipe snat default dump [ verbose ]
```

The optional **verbose** keyword provides more detailed output.

The following command prints the global SNAT settings:

```
bigpipe snat globals show
```

Enabling mirroring for redundant systems

The following example sets SNAT mirroring for all SNAT connections originating at 192.168.225.100:

```
bigpipe snat 192.168.225.100 mirror enable
```

Clearing statistics

You can reset statistics by node or by SNAT address. Use the following syntax to clear all statistics for one or more nodes:

```
bigpipe snat <node ip> [ ...<node ip> ] stats reset
```

Use the following syntax to clear all statistics for one or more SNAT addresses:

```
bigpipe snat <SNAT ip> [ ...<SNAT ip> ] stats reset
```

Use the following command to reset the statistics to zero for the default:

```
bigpipe snat default stats reset
```

timer settings

There are two essential timer settings that you need to configure:

- The node ping timer defines how often the BIG-IP Controller will ping node addresses to verify whether a node is **up** or **down**. It also defines how long the BIG-IP Controller waits for a response from a node before determining that the node is unresponsive and marking the node **down**.
- The idle connection timer defines how long an inactive connection is allowed to remain open before the BIG-IP Controller deletes the record of the connection, closing it and disconnecting the client.

The service check timer is optional, and you need to set it only if you want the BIG-IP Controller to check if a service, or even specific content, is available on a particular node.

The attributes you can configure for timer settings are in Table 1.25.

Attributes	Description
Node ping timer	This timer determines how often the BIG-IP Controller checks to see if nodes are **up** or **down**.
Timer for reaping idle connections	This timer reaps TCP and UDP connections that have been idle for a specified time.
Service check timer	This timer determines whether a server is available by verifying that a particular service is running on a node.
Service checking wildcard ports	Service checking wildcard servers and ports requires you to specify a port, other than port 0, to be service checked.

Table 1.25 *The attributes you can configure for timer settings*

Note

If you plan to use simple service checks, or ECV or EAV service checks, you need to set the service check timer.

Setting the node ping timer

The node ping timer is an essential setting on the BIG-IP Controller that determines how often the BIG-IP Controller checks node addresses to see whether they are **up** and available or **down** and unavailable. The node ping timer setting applies to all nodes configured for use by the BIG-IP Controller, and it is part of the BIG-IP Controller system properties.

The node ping timer sets the amount of time that a server has to respond to a BIG-IP Controller ping in order for the server to be marked **up**. If a server fails to respond within the specified time, the BIG-IP Controller assumes that the server is down, and the BIG-IP Controller no longer sends packets to the services hosted by the server. If the server responds to the next ping, or to subsequent pings, the BIG-IP Controller then marks the server **up**, and resumes sending packets to those services.

◆ Note

*If the Node ping timer (**timeout_node**) interval is shorter than the Service timer (**timeout_svc**) setting, a node can be marked **down** before the services on the node are marked **down**.*

To set the node ping timer using the Configuration utility

1. In the navigation pane, click the BIG-IP Controller icon. The BIG-IP System Properties screen opens.
2. In the Node Ping section of the table, in the **Ping** box, type the frequency (in seconds) at which you want the BIG-IP Controller to ping each node address it manages. A setting of 5 seconds is adequate for most configurations.
3. In the Node Ping section of the table, in the **Timeout** box, type the number of seconds you want the BIG-IP Controller to wait to receive a response to the ping. If the BIG-IP Controller does not receive a response to the ping before the node ping timeout expires, the BIG-IP Controller marks the node **down** and does not use it for load balancing. A setting of 16 seconds is adequate for most configurations.

To set the node ping timer from the command line

To define node ping settings, you use two commands. First, you set the node ping frequency using the **bigpipe tping_node** command, and then you set the node ping timer using the **bigpipe timeout_node** command.

```
bigpipe tping_node <seconds>
bigpipe timeout_node <seconds>
```

For example, the following commands sets the ping frequency at 5 seconds, and the timer to 16 seconds, which should be adequate for most configurations.

```
bigpipe tping_node 5
bigpipe timeout_node 16
```

Displaying the current timeout value

Use the following command to display the current timeout setting for node ping:

```
bigpipe timeout_node show
```

Displaying the current node ping setting

Use the following command to display the current node ping setting:

```
bigpipe tping_node show
```

Setting a timeout value for node ping

Use the following syntax to set the timeout setting for node ping:

```
bigpipe timeout_node <seconds>
```

The sample command below sets the timeout to 33 seconds.

```
bigpipe timeout_node 33
```

Disabling node ping

To disable node ping, you simply set the node ping timeout value to **0** (zero):

```
bigpipe timeout_node 0
```

To turn node ping off, set the **tping_node** interval to **0** seconds:

```
bigpipe tping_node 0
```

WARNING

Node ping is the only form of verification that the BIG-IP Controller uses to determine status of node addresses. If you turn node ping off while one or more node addresses are currently ***down****, the node addresses remain marked* ***down*** *until you turn node ping back on and allow the BIG-IP Controller to verify the node addresses again.*

Setting the timer for reaping idle connections

The BIG-IP Controller supports two timers for reaping idle connections, one for TCP traffic and one for UDP traffic. These timers are essential, and if they are set too high, or not at all, the BIG-IP Controller may run out of memory. Each individual port on the BIG-IP Controller has its own idle connection timer settings.

An idle connection is one in which no data has been received or sent for the number of seconds specified for TCP or UDP connections. For effectively reap idle connections, you should set the idle connection timeout values to be greater than the configured timeouts for the service daemons installed on your nodes.

The TCP idle connection timeout clears the connection tables, avoiding memory problems due to the accumulation of dead, but not terminated, connections.

WARNING

The BIG-IP Controller accepts UDP connections only if you set the UDP idle connection timer.

To set the inactive connection timer in the Configuration utility

1. In the navigation pane, click the expand button (+) next to **Virtual Servers**.
   The Virtual Server tree opens and displays the Ports option.
2. Click **Ports**.
   The Global Virtual Ports screen opens.
3. In the **Port** box, click the port number or service name for which you want to configure the idle connections timeouts.
   The Global Virtual Port screen opens.
4. In the **Idle Connection Timeout TCP** box, type the number of seconds you want to elapse before the BIG-IP Controller drops an idle TCP connection. For HTTP connections, 60 seconds should be adequate, but for other services such as Telnet, higher settings may be necessary.
5. In the **Idle Connection Timeout UDP** box, type the number of seconds you want to elapse before the BIG-IP Controller drops UDP connections.
6. Click the **Apply** button.

To set TCP idle connection timers on the command line

Use the **bigpipe treaper** to define a TCP idle connection timeout for one or more ports at a time. For HTTP connections we recommend only 60 seconds, but for other services such as Telnet we recommend higher settings. The default setting for this timer is 16 minutes (1005 seconds). Use the following syntax for this command.

```
bigpipe treaper <port>... <port> <seconds>
```

For example, the following command sets a 120 second time limit for idle connections on port 443:

```
bigpipe treaper 443 120
```

To set UDP idle connection timers on the command line

You can define a UDP idle connection timeout for one or more ports at a time using the **bigpipe udp** command.

```
bigpipe udp <port>... <port> <seconds>
```

For example, the following command sets a 120-second time limit for idle connections on port 53:

```
bigpipe udp 53 120
```

Setting the service check timer

The service check feature is similar to node ping, but instead of testing the availability of a server, it tests the availability of a particular service running on a server. The service check timer affects the three different types of service checks: simple service check, ECV service check, and EAV service check. To set up simple service check, you need only set the service check timer as described below. To set up ECV service check or EAV service check, however, you need to configure additional settings (see *Extended Content Verification (ECV)*, on page 1-13).

Note that each individual service managed by the BIG-IP Controller has its own service check timer settings.

To set the service check timer in the Configuration utility

1. In the navigation pane, click the expand button (+) next to **Nodes**.
   The Nodes tree opens and displays the Ports option.
2. Click **Ports**.
   The Global Node Ports screen opens.
3. Click the port you want to configure.
   The Global Node Port Properties screen opens.
4. In the **Frequency** box, type the frequency (in seconds) at which you want the BIG-IP Controller to check the service on the node for all defined nodes using this port. Five seconds is adequate for most configurations.
5. In the **Timeout** box, type the number of seconds you want the BIG-IP Controller to wait to receive a response to the service check. If the BIG-IP Controller does not receive a response to the service check before the timeout expires,

the BIG-IP Controller marks the service on the node **down** and does not use it for load balancing. Sixteen (16) seconds is adequate for most configurations.

6. Click the **Apply** button.

To set the service check timer on the command line

To define service check settings, you actually use a series of two commands. First, you set the service check frequency using the **bigpipe tping_svc** command, and then set the service check timer using the **bigpipe timeout_svc** command.

```
bigpipe tping_svc <port> <seconds>
bigpipe timeout_svc <port> <seconds>
```

For example, the following series of commands sets the service check frequency at 5 seconds, and the timer to 16 seconds, which is adequate for most configurations.

```
bigpipe tping_svc 80 5
bigpipe timeout_svc 80 16
```

Service checking for wildcard servers and ports

When you configure a wildcard virtual server with a **0** port using nodes with standard ports, such as 80, with port translation turned **off**, the BIG-IP Controller uses the standard service check timeout values (port 80, for example) to service check the port. For more information about setting the service check timer, see *Setting the service check timer*, on page 1-143.

Using the simple keyword

The simple keyword is being phased out in future releases. This information is provided in order to support existing configurations.

The **simple** keyword is necessary only if you specified a node port of **0**. In previous versions of the BIG-IP Controller, this was the only way to set up a wildcard virtual server that handled

connections for all services. However, we now recommend that you specify a node port and then turn off port translation for the virtual server.

To set up a simple service check for this type of virtual server, add the following entry to the **/etc/bigd.conf** file. Use the following syntax to set a check on a node where the check port is not the node port:

```
simple [<node addr>:]<node port> <check port>
```

For example, a wildcard server is defined with a wildcard port, like this:

```
bigpipe vip 0.0.0.0:0 define n1:0
```

In this case, you must use the **simple** keyword to designate the wildcard <**server:**><**port**> and <**check port**> for the service check:

```
simple n1:0 80
```

Virtual server

Virtual servers provide the ability to map a number of network devices to a single virtual address. A virtual server in combination with a load balancing pool, or rule, provides the ability to load balance connections, use persistence, and also provide high availability features on the BIG-IP Controller.

You must configure a pool of servers before you can create a virtual server that references the pool. Before you configure virtual servers, you need to know:

- If standard virtual servers or wildcard virtual servers meet the needs of your network
- Whether you need to activate optional virtual server properties

Once you know which virtual server options are useful in your network, you can:

- Define standard virtual servers
- Define wildcard virtual servers

The attributes you can configure for a virtual server are in Table 1.26.

Attributes	Description
Standard virtual server	A standard virtual server sends connection requests to load balancing pools or rules.
Wildcard virtual server	A wildcard virtual server is typically used to make requests to hosts on the internet from a network behind the BIG-IP Controller.
Network virtual server	A network virtual server handles a whole range of addresses in a network.
Other virtual server attributes	You can set connection limits, translation properties, last hop pools, and mirroring information for virtual servers.

Table 1.26 *The attributes you can configure for a virtual server*

Using standard or wildcard virtual servers

Virtual servers reference a pool you create that contains a group of content servers, firewalls, routers, or cache servers, and they are associated with one or more external interfaces on the BIG-IP Controller.

You can configure two different types of virtual servers:

- **Standard virtual servers**
  A standard virtual server represents a site, such as a web site or an FTP site, and it provides load balancing for a pool of content servers or other network devices. The virtual server IP address should be the same IP address that you register with DNS for the site that the virtual server represents.
- **Wildcard virtual servers**
  A wildcard virtual server load balances a pool of transparent network devices such as firewalls, routers, or cache servers. Wildcard virtual servers are configured with an IP address of 0.0.0.0, and sometimes with a virtual port of 0.

Note that both the Configuration utility and the BIG/pipe command line utility accept host names in place of IP addresses, and also accept standard service names in place of port numbers.

Defining virtual servers

A standard virtual server represents a specific site, such as an Internet web site or an FTP site, and it load balances content servers that are members of a pool. The IP address that you use for a standard virtual server should match the IP address that DNS associates with the site's domain name.

◆ Note

If you are using a 3-DNS Controller in conjunction with the BIG-IP Controller, the 3-DNS Controller uses the IP address associated with the registered domain name in its own configuration. For details, refer to the ***3-DNS Controller Administrator Guide****.*

To define a standard virtual server that references a pool in the Configuration utility

1. In the navigation pane, click **Virtual Servers**.
2. On the toolbar, click **Add Virtual Server**.
   The Add Virtual Server screen opens.
3. In the **Address** box, enter the virtual server's IP address or host name.
4. In the **Netmask** box, type an optional netmask. If you leave this setting blank, the BIG-IP Controller uses a default netmask based on the IP address you entered for the virtual server. Use the default netmask unless your configuration requires a different netmask.
5. In the **Broadcast** box, type the broadcast address for this virtual server. If you leave this box blank, the BIG-IP Controller generates a default broadcast address based on the IP address and netmask of this virtual server.
6. In the **Port** box, either type a port number, or select a service name from the drop-down list.
7. For **Interface**, select the external (destination processing) interface on which you want to create the virtual server. Select **default** to allow the Configuration utility to select the interface based on the network address of the virtual server. If no external interface is found for that network, the virtual server is created on the first external interface. If you choose **None**, the BIG-IP Controller does not create an alias and generates no ARPs for the virtual IP address. In this case, the BIG-IP Controller accepts traffic on all interfaces.
8. In Resources, click the **Pool** button.
   If you want to assign a load balancing rule to the virtual server, click **Rule** and select a rule you have configured.
9. In the Pool list, select the pool you want to apply to the virtual server.
10. Click the **Apply** button.

To define a standard virtual server mapping on the command line

Type the **bigpipe vip** command as shown below. Also, you can use host names in place of IP addresses, and that you can use standard service names in place of port numbers.

```
bigpipe vip <virt IP>:<port> use pool <pool_name>
bigpipe vip <virt IP>:<port> use rule <rule_name>
```

For example, the following command defines a virtual server that maps to the pool **my_pool**:

```
bigpipe vip 192.200.100.25:80 use pool my_pool
```

Defining wildcard virtual servers

Wildcard virtual servers are a special type of virtual server designed to manage network traffic for transparent network devices, such as transparent firewalls, routers, proxy servers, or cache servers. A wildcard virtual server manages network traffic that has a destination IP address unknown to the BIG-IP Controller. A standard virtual server typically represents a specific site, such as an Internet web site, and its IP address matches the IP address that DNS associates with the site's domain name. When the BIG-IP Controller receives a connection request for that site, the BIG-IP Controller recognizes that the client's destination IP address matches the IP address of the virtual server, and it subsequently forwards the client to one of the content servers that the virtual server load balances.

However, when you are load balancing transparent nodes, a client's destination IP address is going to seem random. The client is connecting to an IP address on the other side of the firewall, router, or proxy server. In this situation, the BIG-IP Controller cannot match the client's destination IP address to a virtual server IP address. Wildcard virtual servers resolve this problem by not translating the incoming IP address at the virtual server level on the BIG-IP Controller. For example, when the BIG-IP Controller does not find a specific virtual server match for a client's destination IP address, it matches the client's IP address to a wildcard virtual server. The BIG-IP Controller then forwards the client's packet to

one of the firewalls or routers that the wildcard virtual server load balances, which in turn forwards the client's packet to the actual destination IP address.

A note about wildcard ports

When you configure wildcard virtual servers and the nodes that they load balance, you can use a wildcard port (port **0**) in place of a real port number or service name. A wildcard port handles any and all types of network services.

A wildcard virtual server that uses port **0** is referred to as a ***default wildcard virtual server***, and it handles traffic for all services. A ***port-specific wildcard virtual server*** handles traffic only for a particular service, and you define it using a service name or a port number. If you use both a default wildcard virtual server and port-specific wildcard virtual servers, any traffic that does not match either a standard virtual server or one of the port-specific wildcard virtual servers is handled by the default wildcard virtual server.

You can use port-specific wildcard virtual servers for tracking statistics for a particular type of network traffic, or for routing outgoing traffic, such as HTTP traffic, directly to a cache server rather than a firewall or router.

We recommend that when you define transparent nodes that need to handle more than one type of service, such as a firewall or a router, you specify an actual port for the node and turn off port translation for the virtual server.

◆ Note

*When you define a virtual server with port translation turned **off**, and you want to perform a service check on that node, you must configure service check intervals and timeouts using the port specified for the node. Then you can configure a service check. See Service checking for wildcard servers and ports, on page 1-144, for more details.*

Defining the wildcard virtual server mappings

There are two procedures required to set up a wildcard virtual server. First, you must define the wildcard virtual server. Then you must turn port translation off for the virtual server.

To define a wildcard virtual server mapping using the Configuration utility

1. In the navigation pane, click **Virtual Servers**.
2. On the toolbar, click **Add Virtual Server**.
   The Add Virtual Server screen opens.
3. In the **Address** box, type the wildcard IP address **0.0.0.0**.
4. In the **Netmask** box, type an optional netmask.
   If you leave this box blank, the BIG-IP Controller generates a default netmask address based on the IP address of this virtual server. Use the default netmask unless your configuration requires a different netmask.
5. In the **Broadcast** box, type the broadcast address for this virtual server.
   If you leave this box blank, the BIG-IP Controller generates a default broadcast address based on the IP address and netmask of this virtual server.
6. In the **Port** box, type a port number, or select a service name from the drop-down list. Note that port **0** defines a wildcard virtual server that handles all types of services. If you specify a port number, you create a port-specific wildcard virtual server. The wildcard virtual server only handles traffic for the port specified.
7. For **Interface**, select the external (destination processing) interface on which you want to create the virtual server.
   If you choose **None**, the BIG-IP Controller does not create an alias and generates no ARPs for the virtual IP address (see the ***BIG-IP Controller Administrator Guide***, *Optimizing large configurations* for details).
8. In Resources, click the **Pool** button.

9. In the Pool list, select the pool you want to apply to the virtual server.
10. Click the **Apply** button.

To turn off port translation for a wildcard virtual server in the Configuration utility

After you define the wildcard virtual server with a wildcard port, you must disable port translation for the virtual server.

1. In the navigation pane, click **Virtual Servers**.
   The Virtual Servers screen opens.
2. In the virtual server list, click the virtual server for which you want to turn off port translation.
   The Virtual Server Properties screen opens.
3. In the Enable Translation section, clear the **Port** box.
4. Click the **Apply** button.

To define a wildcard virtual server mapping on the command line

There are three commands required to set up a wildcard virtual server. First, you must define a pool that contains the addresses of the transparent devices. Next, you must define the wildcard virtual server. Then you must turn port translation off for the virtual server. To define the pool of transparent devices, use the **bigpipe pool** command. For example, you can create a pool of transparent devices called **transparent_pool** that uses the Round Robin load balancing mode:

```
bigpipe pool transparent_pool { lb_mode rr member
   <member_definition>... member <member_definition> }
```

To define the virtual server, use the **bigpipe vip** command:

```
bigpipe vip <virtual IP>:<port> use pool <pool_name>
```

After you define the virtual server, you can enable or disable port translation using the following command:

```
bigpipe vip <virtual IP>:<port> translate port enable | disable
```

For example, you can create a pool of transparent devices called **transparent_pool** that uses the Round Robin load balancing mode:

```
bigpipe pool transparent_pool { lb_mode rr member 10.10.10.101:80
   member 10.10.10.102:80 member 10.10.10.103:80 }
```

After you create the pool of transparent nodes, use the following command to create a wildcard virtual server that maps to the pool **transparent_pool**. Because the members are firewalls and need to handle a variety of services, the virtual server is defined using port **0** (or ***** or **any**). You can specify any valid non-zero port for the node port and then turn off port translation for that port. In this example, service checks ping port 80.

```
bigpipe vip 0.0.0.0:0 use pool transparent_pool
```

After you define the virtual server, turn off port translation for the port in the virtual server definition. In this example, port 80 is used for service checking. If you do not turn off port translation, all incoming traffic would be translated to port 80.

```
bigpipe vip 0.0.0.0:0 translate port disable
```

Configuring a network virtual server

You can configure a network virtual server to handle a whole network range, instead of just one IP address, or all IP addresses (wildcard virtual servers). For example, the virtual server in Figure 1.18 handles all traffic addresses in the 192.168.1.0 network:

```
bigpipe vip 192.168.1.0:0 none {
    netmask 255.255.255.0 broadcast 192.168.1.255
    use pool ingress_firewalls
}
```

Figure 1.18 *A sample network virtual server*

◆ Note

Network virtual servers should be assigned to interface ***none****.*

A network virtual server is a virtual server that has no bits set in the host portion of the IP address. In other words, the host portion is zero. You must specify a network mask to indicate which portion of the address is the network address and which portion is the host address. In the previous example, since the network mask is 255.255.255.0, the network portion of the address is **192.168.1** and the host portion is **.0**. The previous example would direct all traffic destined to the subnet 192.168.1.0/24 through the BIG-IP Controller to the **ingress_firewalls** pool.

Another way you can use this feature is to create a catch-all webserver for an entire subnet. For example, you could create the following network virtual server (Figure 1.19).

```
bigpipe vip 192.168.1.0:http none {
    netmask 255.255.255.0 broadcast 192.168.1.255
    use pool default_webservers
}
```

Figure 1.19 *A catch-all web server configuration.*

This configuration directs a web connection destined to any address within the subnet 192.168.1.0/24 to the **default_webservers** pool.

Displaying information about virtual servers

Use the following syntax to display information about all virtual servers included in the configuration:

```
bigpipe vip show
```

Use the following syntax to display information about one or more virtual servers included in the configuration:

```
bigpipe vip <virt ip>:<port> [...<virt ip>:<port>] show
```

The command displays information such as the nodes associated with each virtual server, the nodes' status, and the current, total, and maximum number of connections managed by the virtual server since the BIG-IP Controller was last rebooted.

Defining an interface for a virtual server

If you have multiple external (destination processing) interfaces, you can specify one of them when you define a virtual server. If you specify an interface name, the BIG-IP Controller responds to ARP requests for the virtual address on that interface. If you do not specify an interface name, the BIG-IP Controller responds to ARP requests for the virtual server on the default interface. If you do not want the BIG-IP Controller to respond to ARP requests on any interface, use the option **none** in place of the an <**ifname**> parameter.

All virtual servers that share a virtual address must use the same external interface. Changing the interface for a virtual server changes the interface for all virtual servers having the same virtual address.

Setting a user-defined netmask and broadcast

The default netmask for a virtual address, and for each virtual server hosted by that virtual address, is determined by the network class of the IP address entered for the virtual server. The default broadcast is automatically determined by the BIG-IP Controller, and it is based on the virtual address and the current netmask. You can override the default netmask and broadcast for any virtual address.

All virtual servers hosted by the virtual address use the netmask and broadcast of the virtual address, whether they are default values or they are user-defined values.

Note that if you want to use a custom netmask and broadcast, you define both when you define the virtual server:

```
bigpipe vip <virt ip>[:<port>] [<ifname>] [netmask <ip>] \
   [broadcast <ip>] use pool <pool_name>
```

◆ Note

The BIG-IP Controller calculates the broadcast based on the IP address and the netmask. A user-defined broadcast address is not necessary.

Again, even when you define a custom netmask and broadcast in a specific virtual server definition, the settings apply to all virtual servers that use the same virtual address. The following sample command shows a user-defined netmask and broadcast:

```
bigpipe vip www.SiteOne.com:http netmask 255.255.0.0 \
   broadcast 10.0.140.255 use pool my_pool
```

The **/bitmask** option shown in the following example applies network and broadcast address masks. In this example, a 24-bit bitmask sets the network mask and broadcast address for the virtual server:

```
bigpipe vip 206.168.225.1:80/24 use pool my_pool
```

You can generate the same broadcast address by applying the 255.255.255.0 netmask. The effect of the bitmask is the same as applying the 255.255.255.0 netmask. The broadcast address is derived as 206.168.225.255 from the network mask for this virtual server.

Setting a connection limit

The default setting is to have no limit to the number of concurrent connections allowed on a virtual server. You can set a concurrent connection limit on one or more virtual servers using the following command:

```
bigpipe vip <virt ip>[:<port>] [...<virt ip>[:<port>] ] limit \
   <max conn>
```

The following example shows two virtual servers set to have a concurrent connection limit of 5000 each:

```
bigpipe vip www.SiteOne.com:http www.SiteTwo.com:ssl limit 5000
```

To turn the limit off, set the <**max conn**> variable to zero:

```
bigpipe vip <virt ip>[:<port>] [...<virt ip>[:<port>] ] limit 0
```

Setting translation properties for virtual addresses and ports

Turning port translation off for a virtual server is useful if you want to use the virtual server to load balance connections to any service. Use the following syntax to enable or disable port translation for a virtual server.

```
bigpipe vip <virt ip>:<port> translate port enable | disable | show
```

You can also configure the translation properties for a virtual server address. This option is useful when the BIG-IP Controller is load balancing devices which have the same IP address. This is typical with the nPath routing configuration where duplicate IP addresses are configured on the loopback device of several servers. Use the following syntax to enable or disable address translation for a virtual server.

```
bigpipe vip <virt ip>:<port> translate addr enable | disable | show
```

Setting up last hop pools for virtual servers

In cases where you have more than one router sending connections to a BIG-IP redundant system, you may want to route connections back through the same router from which they were received. To configure a last hop pool, you must first create a pool that contains the routers for the BIG-IP redundant system. After you create a router pool, use the following syntax to configure a last hop pool for a virtual server.

```
bigpipe vip <virt ip>:<port> lasthop pool <pool_name> | none | show
```

Mirroring connection information

Mirroring provides seamless recovery for current connections and when a BIG-IP Controller fails. When you use the mirroring feature, the peer controller maintains the same current connection and persistence information as its partner controller. Transactions such as FTP file transfers continue as though uninterrupted.

To control mirroring for a virtual server, use the **mirror** command to enable or disable mirroring of connections. The syntax of the command is:

```
bigpipe vip <virt ip>:<port> mirror conn enable | disable
```

To print the current mirroring setting for a virtual server:

```
bigpipe vip <virt ip>:<port> mirror conn show
```

If you do not specify **conn**, the BIG-IP Controller displays all mirrored connection information.

◆ Note

If you set up mirroring on a virtual server that supports FTP connections, you need to mirror the control port virtual server, and the data port virtual server.

The following example shows the two commands used to enable mirroring for virtual server **v1** on the FTP control and data ports:

```
bigpipe vip v1:21 mirror conn enable
bigpipe vip v1:20 mirror conn enable
```

Removing and returning a virtual server to service

You can remove an existing virtual server from network service, or return the virtual server to service, using the **disable** and **enable** keywords. When you disable a virtual server, the virtual server no longer accepts new connection requests, but it allows current connections to finish processing before the virtual server goes **down**.

Use the following syntax to remove a virtual server from network service:

```
bigpipe vip <virt ip>:<port> [...<virt ip>:<port>] disable
```

Use the following syntax to return a virtual server to network service:

```
bigpipe vip <virt ip>:<port> enable
```

Removing and returning a virtual address to service

You can remove an existing virtual address from network service, or return the virtual address to service, using the **disable** and **enable** keywords. Note that when you enable or disable a virtual address, you inherently enable or disable all of the virtual servers that use the virtual address.

```
bigpipe vip <virt ip> disable
```

Use the following syntax to return a virtual address to network service:

```
bigpipe vip <virt ip> enable
```

Displaying information about virtual addresses

You can also display information about the virtual addresses that host individual virtual servers. Use the following syntax to display information about one or more virtual addresses included in the configuration:

```
bigpipe vip <virt ip> [... <virt ip> ] show
```

The command displays information such as the virtual servers associated with each virtual address, the status, and the current, total, and maximum number of connections managed by the virtual address since the BIG-IP Controller was last rebooted, or since the BIG-IP Controller became the active unit (redundant configurations only).

Deleting a virtual server

Use the following syntax to permanently delete one or more virtual servers from the BIG-IP Controller configuration:

```
bigpipe vip <virt ip>:<port> [... <virt ip>:<port>] delete
```

Resetting statistics for a virtual server

Use the following command to reset the statistics for an individual virtual server:

```
bigpipe vip [<vip ip:port>] stats reset
```

Turning software acceleration off for virtual servers using IPFW rate filters

Additional enhancements are included in this release that speed packet flow for TCP connections when the packets are not fragmented. In most configurations these software enhancements are automatically turned on and do not require any additional configuration.

However, you may want to turn off these enhancements for individual virtual servers that use IPFW rate filters. With the speed enhancements on, IPFW only examines the first SYN packet in any given connection. If want to filter all packets, you should turn the speed enhancements off. To do this, you must first set the global state of the system on, and then you must turn the feature off for individual virtual servers that use IPFW rate filtering. You can change the settings for these enhancements from the command line or in the Configuration utility.

Setting software acceleration controls from the command line

Before you can turn off software acceleration for a virtual server, you must set the **sysctl** variable **bigip.fastpath_active** to on (2) with the following command:

```
sysctl -w bigip.fastpath_active=2
```

After you set the **sysctl** variable, use the following **bigpipe** commands to disable software acceleration for existing virtual servers that use IPFW rate filtering:

```
bigpipe vip <ip>:<port> accelerate disable
```

For example, if you want to turn acceleration off for the virtual server 10.10.10.50:80, type the following command:

```
bigpipe vip 10.10.10.50:80 accelerate disable
```

You can define a virtual server with acceleration disabled using the following syntax:

```
bigpipe vip <ip>:<port> use pool the_pool
   accelerate disable
```

For example, if you want to define the virtual server 10.10.10.50:80 with the pool **IPFW_pool** and acceleration turned off, type the following command:

```
bigpipe vip 10.10.10.50:80 use pool IPFW_pool
   accelerate disable
```

Using additional features with virtual servers

After you create a pool and define a virtual server that references the pool, you can set up additional features, such as network address translation (NAT) or extended content verification (ECV). If you are planning on using any of these features, you may want to read the corresponding section before you actually begin the virtual server configuration process:

- Network address translations (see *NAT*, on page 1-53)
- Extended Content Verification service checking (see *Extended Content Verification (ECV)*, on page 1-13)
- Persistence for connections that should return to the node to which they last connected (see *Setting up persistence for a pool*, on page 1-65).

F5 Networks BIG-IP Local Traffic Controller

2

bigpipe Command Reference

bigpipe commands

This chapter lists the various **bigpipe** commands with descriptions. At the end of the chapter is a list of commands from previous versions of the **bigpipe** utility.

Table 2.1 explains the conventions used in the command line syntax described in this chapter.

Item in text	Description
****	Continue to the next line without typing a line break.
< >	You enter text for the enclosed item. For example, if the command has **<your name>**, type in your name.
|	Separates alternate options for a command.
[ ]	Syntax inside the square brackets is optional.
...	Indicates that you can type a series of items.

Table 2.1 *Command line conventions used in this manual*

Table 3.2 provides a concise listing of the commands discussed in this chapter, along with the page where you can find more details. Refer to the command entry pages for sample syntax and a description of how to use the command. Some entries contain additional information about using the command.

Command	Description	Page
-?	Displays online help for an individual **bigpipe** command.	2-4
alias	Defines an IP alias to be pinged on behalf of a specific group of nodes.	2-5
configsync	Synchronizes the **/etc/bigip.conf** between the two BIG-IP Controller units in a redundant system.	2-7
conn	Shows information about current connections such as the source IP address, virtual server and port, and node.	2-8
-d	Verifies command syntax for the specified command without executing a command.	2-9
-f	Resets the BIG-IP Controller and loads a specified configuration file.	2-10
failover	Sets the BIG-IP Controller as active or standby.	2-11
gateway	Turns the gateway fail-safe feature on and off.	2-12
-h and -help	Displays online help for **bigpipe** command syntax.	2-13
interface	Sets options on individual interfaces.	2-14
ipalias	Configure shared addresses on interfaces.	2-21
-l	Loads the BIG-IP Controller configuration without resetting the current configuration.	2-22
lb	Sets the load balancing mode.	2-23
maint	Toggles the BIG-IP Controller into and out of maintenance mode.	2-25
mirror	Sets mirroring of the active BIG-IP Controller to the standby controller.	2-26
-n	Displays ports numerically rather than by service name	2-30
nat	Defines external network address translations for nodes.	2-27
node	Defines node property settings.	2-31
persist	Defines and displays persistence settings for simple TCP and UDP persistence.	2-35
pool	Defines load balancing pools.	2-36
port	Defines properties for virtual ports.	2-44
proxy	Defines the properties of the SSL gateway for the SSL Accelerator.	2-46
-r	Clears the BIG-IP Controller configuration and counter values.	2-50
ratio	Sets load-balancing weights and priority levels used in the Ratio and Priority load balancing modes.	2-51

Command	Description	Page
rule	Defines load balancing rules.	2-53
-s	Writes the current configuration to a file.	2-57
snat	Defines and sets options for SNAT (Secure NAT).	2-58
summary	Displays summary statistics for the BIG-IP Controller.	2-62
timeout_node	Sets the amount of time that node addresses have to respond to a ping issued by the BIG-IP Controller.	2-64
timeout_svc	Sets the amount of time that services have to respond to a service check issued by the BIG-IP Controller.	2-66
tping_node	Sets the interval at which the BIG-IP Controller pings node addresses to determine node status.	2-68
tping_svc	Sets the interval at which the BIG-IP Controller issues service checks to nodes to determine node status.	2-69
treaper	Sets the timeout for idle TCP connections on ports.	2-71
udp	Enables UDP traffic on ports, and sets the timeout for idle UDP connections.	2-73
unit	Displays the unit number assigned to a particular BIG-IP Controller.	2-75
-v	Displays the **bigpipe** utility version number.	2-76
version	Displays the BIG-IP Controller software version number.	2-77
vip	Defines virtual servers, virtual server mappings, and virtual server properties.	2-78

Table 2.2 *A list of* ***bigpipe*** *commands discussed in this chapter.*

On page 2-87 you can find a list of backward-compatible commands from previous versions of the BIG-IP Controller that are compatible with this version.

-?

bigpipe <command> -?

Description

For certain commands, displays online help, including complete syntax, description, and other related information. For example, to see online help for the **bigpipe port** command, type:

bigpipe port -?

alias

```
bigpipe alias [<node ip> [...<node ip>] ] show
bigpipe alias <node ip> [...<node ip>] delete
bigpipe alias <node ip> [...<node ip>] pingnode <pingnode ip>
```

Description

Defines a single node address to represent a group of node addresses which are actually IP aliases on the same physical server. To determine if the nodes associated with the representative node alias are available, the BIG-IP Controller sends a single node ping to the node alias, rather than an individual ping to each node address.

Note that you may also find this feature useful for nodes that are configured for service check, as long as each node uses the same port number.

Defining a node alias

Use the following syntax to define the node alias for one or more node addresses, where <**pingnode ip**> is the node alias (the node address that represents the group):

```
bigpipe alias <node ip> [...<node ip>] pingnode <pingnode ip>
```

◆ Note

*The address that serves as the node alias (<**pingnode ip**>) must be a node address that is already defined in one or more virtual server mappings.*

The following command defines a node alias for two node addresses, 192.168.42.2 and 192.168.42.3. The BIG-IP Controller performs node pings on the alias address 192.168.42.1 to determine the availability of 192.168.42.2 and 192.168.42.3.

```
bigpipe alias 192.168.42.2 192.168.42.3 pingnode 192.168.42.1
```

Deleting a node alias

The following command deletes the node alias defined for the specific node:

```
bigpipe alias <node ip> delete
```

Displaying current node aliases

The following command displays all node aliases defined on the BIG-IP Controller:

```
bigpipe alias show
```

The following command displays the node alias defined for a specific node:

```
bigpipe alias <node ip> show
```

configsync

bigpipe configsync [all]

Description

Synchronizes configurations of two BIG-IP Controllers in a redundant system by copying the configuration file(s) from the active unit to the standby unit.

Using the **configsync** command without the **all** option synchronizes only the boot configuration file **/etc/bigip.conf**.

The **all** option changes the set of configuration files that are modified when the command is executed. When you synchronize a configuration using **configsync all** command, the following configuration files are copied to the other BIG-IP Controller:

- **The common BIG/db keys**
- **/etc/bigip.conf**
- **/etc/bigd.conf**
- **/etc/hosts.allow**
- **/etc/hosts.deny**
- **/etc/ipfw.conf**
- **/etc/rateclass.conf**
- **/etc/ipfwrate.conf**
- **/etc/snmpd.conf**
- **rc.sysctl**

Be sure to save the current configuration to the **/etc/bigip.conf** file before you use the configuration synchronization feature.

WARNING

If you are synchronizing a standby controller that already has configuration information defined, we recommend that you back up that controller's original configuration file(s) first.

conn

bigpipe conn [<virt ip>[:<port>]] dump [mirror]

Description

Displays information about current client connections to virtual addresses and virtual servers.

The following command displays all current client connections:

bigpipe conn dump

The output shows the source IP, virtual server and port, and node connected to.

```
bigip conn dump

from                      vip                      node
100.100.100.30:49152 ->  100.100.100.100:23 -> 200.200.200.10:23
100.100.101.90:49153 ->  100.100.100.100:80 -> 200.200.200.10:80
...
```

Figure 2.1 *Formatted output of the* ***conn*** *command*

This command can also show connections that are active on the given controller as well as those that are standby connections for the peer BIG-IP Controller. By default, the **dump** command only shows items that are active on the given unit. To see standby items, you must use the **mirror** qualifier.

bigpipe conn dump mirror

-d

```
bigpipe -d [-]
bigpipe -d -f <filename>
```

Description

Parses the command line and checks syntax without executing the specified command.

This distinguishes between valid and invalid commands, and is particularly useful with the **-f** option, to validate the configuration file.

Use the **-d** command followed by a command that you want to validate:

```
bigpipe -d vip 10.10.10.100:80 use pool my_pool
```

The command checks the syntax and logic, reporting any errors that would be encountered if the command executed.

Use the **-d** command together with the **-f <filename>** command to validate the specified configuration file. For example, to check the syntax of the configuration file **/etc/altbigpipe.conf**, use the following command:

```
bigpipe -d -f /etc/altbigip.conf
```

-f

bigpipe -f <filename>

Description

Resets all of the BIG-IP Controller settings and then loads the configuration settings from the specified file, typically the **/etc/bigip.conf** file, or another file that you specify.

```
bigpipe -f /etc/bigip.conf
```

For testing purposes, you can save a test configuration by renaming it to avoid confusion with the boot configuration file. To load a test configuration, use the **-f** command with the <**filename**> parameter. For example, if you renamed your configuration file to **/etc/bigtest.conf**, the command would be:

```
bigpipe -f /etc/bigtest.conf
```

failover

`bigpipe failover standby | show | init | failback`

Description

This group of commands affects the fail-over status of the BIG-IP Controller.

In an active/standby or active-active configuration, run the following command to place a controller in standby mode:

```
bigpipe failover standby
```

Show the status of the controller with the following command:

```
bigpipe failover show
```

◆ Note

*The **failback** command is only applicable if you are running a redundant system in active-active mode.*

In an active-active configuration, run the following command after you issue the **bigpipe failover standby** command. This allows the inactive controller to resume handling connections:

```
bigpipe failover failback
```

You can use the **bigpipe failover init** command to refresh the parameters of the fail-over daemon (**/sbin/sod**) with any new configuration data entered in the BIG/db database.

```
bigpipe failover init
```

gateway

bigpipe gateway failsafe arm | disarm | show

Description

Turns the gateway fail-safe feature on and off. This command is supported only for redundant systems. To configure gateway pingers, you must first set the IP address of the router, ping interval, and timeout period in BIG/db. For information about configuring gateway fail-safe, see the ***BIG-IP Controller Administrator Guide***, *Working with Advanced Redundant System Features*.

The typical use of gateway fail-safe is a setup where active and standby BIG-IP Controllers use different routers as gateways to the internet. Fail-over is triggered if the gateway for the active controller is unreachable. Note that this is not a condition that is reliably detected by the interface fail-safe feature, but is reliably detected by gateway fail-safe.

To arm fail-safe on the gateway, enter the following command:

```
bigpipe gateway failsafe arm
```

To disarm fail-safe on the gateway, enter the following command:

```
bigpipe gateway failsafe disarm
```

To see the current fail-safe status for the gateway, enter the following command:

```
bigpipe gateway failsafe show
```

-h and -help

bigpipe [-h | -help]

Description

Displays the **bigpipe** command syntax or usage text for all current commands.

◆ Note

*More detailed man pages are available for some individual **bigpipe** commands. To display detailed online help for the **bigpipe** command, type: **man bigpipe**.*

interface

```
bigpipe interface <ifname> show
bigpipe interface <ifname> source enable | disable
bigpipe interface <ifname> dest enable | disable
bigpipe interface <interface> source_translation \
   [ enable | disable ]
bigpipe interface <ifname> adminport open | lockdown
bigpipe interface <ifname> failsafe arm | disarm | show
bigpipe interface <ifname> timeout <seconds> | show
bigpipe interface <ifname> mac_masq <mac_addr> | show
bigpipe interface <ifname> vlans enable | disable | show
```

Description

Displays names of installed network interface cards and allows you to set properties for each network interface card.

◆ Note

Interface fail-safe is not designed for gateway or node failure detection, as it cannot detect router or node failures in instances where other sources of Ethernet traffic are active on the interface.

Designating an internal or external interface

With BIG-IP Controller version 3.0 and later, you can define interfaces using three new parameters: **source**, **dest**, and **adminport**. You can mix and match these options to streamline the performance of the BIG-IP Controllers in the network. Table 2.3 describes the attributes that determine the way an interface handles connections.

Interface type	Attributes
Internal	Process source addresses Administrative ports open
External	Process destination addresses Administrative ports locked down

Table 2.3 *Attributes of internal and external interfaces*

Use the following syntax to designate an interface as an internal or external interface.

```
bigpipe interface <ifname> source enable | disable
bigpipe interface <ifname> dest enable | disable
bigpipe interface <ifname> adminport open | lockdown
```

The <**ifname**> parameter takes a valid interface name such as:

- **exp0**
  This is the first Intel NIC
- **fpa1**
  This is the second FDDI NIC
- **de2**
  This is the third DEC/SMC NIC
- **sk0**
  This is the first SysKonnect Gigabit Ethernet NIC

◆ **Note**

Dual port Ethernet NICs show up as two distinct interfaces

The following sample syntax configures the interface **exp0** as an internal interface on the BIG-IP Controller:

```
bigpipe interface exp0 source enable
bigpipe interface exp0 dest disable
bigpipe interface exp0 adminport open
```

The following sample syntax configures the interface **exp1** as an external interface on the BIG-IP Controller:

```
bigpipe interface exp1 source disable
bigpipe interface exp1 dest enable
bigpipe interface exp1 adminport lockdown
```

WARNING

Use caution when redefining interfaces. When you reconfigure interfaces, make sure that you have set up the interfaces you need for operation. It is possible to accidentally take the controller out of network service by redefining interfaces.

Source translation processing

When source translation processing is enabled on an interface, then the BIG-IP Controller processes packets arriving at the interface when those packets are coming from a node, SNAT, or NAT internal address. In this situation, the interface rewrites the source address of the IP packet, changing it from the real server's IP address, or original NAT address, to the virtual server or translated NAT address, respectively. Also, when the last hop feature is enabled on a virtual server, the packet is routed back to the network device that first transmitted the connection request to the virtual server.

To configure source and destination processing from the command line

Use the following syntax to configure source and destination processing on the specified interface:

```
bigpipe interface <interface> source_translation [
    enable | disable ]
```

Displaying status for interfaces

Use the following syntax to display the current status and the settings for all installed interface cards:

bigpipe interface show

Figure 2.2 is an example of the output you see when you issue this command on an active/standby controller in active mode.

```
exp0         11.11.11.2, dest enable, source disable, disarmed, timeout 30
 shared alias 11.11.11.3  netmask 255.0.0.0  broadcast 11.255.255.255  unit 1
exp1         11.12.11.2, dest disable, source enable, disarmed, timeout 30
 shared alias 11.12.11.3  netmask 255.0.0.0  broadcast 11.255.255.255  unit 1
```

Figure 2.2 *The* ***bigpipe interface show*** *command output*

Use the following syntax to display the current status and the setting for a specific interface.

bigpipe interface <ifname> show

Arming and disarming the fail-safe mode

Use the following command to activate the BIG-IP Controller interface fail-safe mode.

bigpipe interface <ifname> failsafe arm

When armed, the active controller automatically fails over to the standby controller whenever the active controller detects that there is no activity on the specified interface, and subsequently detects no activity on the interface in response to ARP requests. The default fail-safe mode is set to **disarm**.

WARNING

You should arm the fail-safe mode only after you configure the BIG-IP Controller, and both the active and standby units are ready to be placed into a production environment.

Before using the **bigpipe interface failsafe** command, you must specify the default route in the **/etc/hosts** and **/etc/netstart** files.

Use the following command to deactivate the BIG-IP Controller interface fail-safe mode.

```
bigpipe interface <ifname> failsafe disarm
```

Setting the fail-safe timeout

To set the amount of time (in seconds) that an interface is monitored for activity in response to a BIG-IP Controller ARP request, use the following syntax:

```
bigpipe interface <ifname> timeout <seconds>
```

If no activity is detected on the interface within the specified time, the BIG-IP Controller assumes that the interface is down. Note that the default setting is 30 seconds.

Warning messages and ARP requests are generated after half of the specified time-out period. In the case of an armed BIG-IP Controller in a BIG-IP redundant system, traffic is switched from the active unit to the standby unit at the end of the time-out period. Note that the fail-safe timeout is used only if the fail-safe option is armed on the interface.

Viewing the timeout setting

Use the following syntax to view the fail-over timeout setting for a specific interface:

```
bigpipe interface <ifname> timeout show
```

Displaying the current fail-safe status

Use the following syntax to display the current status and settings for the BIG-IP Controller fail-safe mode:

```
bigpipe interface failsafe show
```

Setting the MAC masquerade address

Sharing the MAC masquerade address makes it possible to use BIG-IP Controllers in a network topology using secure hubs. You can view the media access control (MAC) address on a given controller using the following command:

```
/sbin/ifconfig -a
```

Use the following syntax to set the MAC masquerade address that will be shared by both BIG-IP Controllers in the redundant system.

```
bigpipe interface <ifname> mac_masq <MAC addr>
```

WARNING

*You must specify a default route before using the **mac_masq** command. You specify the default route in the **/etc/hosts** and **/etc/netstart** files.*

Find the MAC address on both the active and standby units and choose one that is similar but unique. A safe technique for choosing the shared MAC address follows.

Suppose you want to set up **mac_masq** on the external interfaces. Using the **ifconfig -a** command on the active and standby units, you note that their MAC addresses are:

```
Active: exp0 = 0:0:0:ac:4c:a2
Standby: exp0 = 0:0:0:ad:4d:f3
```

In order to avoid packet collisions, you now must choose a unique MAC address. The safest way to do this is to select one of the addresses and logically **OR** the first byte with **0x40**. This makes the MAC address a locally administered MAC address.

In this example, either 40:0:0:ac:4c:a2 or 40:0:0:ad:4d:f3 would be a suitable shared MAC address to use on both BIG-IP Controllers in the redundant system.

The shared MAC address is used only when the BIG-IP Controller is in active mode. When the unit is in standby mode, the original MAC address of the network card is used.

If you do not configure **mac_masq** on startup, or when transitioning from standby mode to active mode, the BIG-IP Controller sends gratuitous ARP requests to notify the default router and other machines on the local Ethernet segment that its MAC address has changed. See RFC 826 for more details on ARP.

Note

You can use the same technique to configure a shared MAC address for each interface.

Enabling VLAN communication for an interface

To use IEEE 802.1q VLAN Trunk mode, you must first set up VLAN tags in **/etc/netstart** and in the shared IP in BIG/db. For detailed information about setting up VLAN tags, see the ***BIG-IP Controller Administrator Guide***, *Using Advanced Network Configurations*.

Use the following syntax to enable, disable, or show the VLAN status of the specified internal interface:

```
bigpipe interface <ifname> vlans enable | disable | show
```

ipalias

```
ipalias <ifname> <if address> netmask <ip mask> [ broadcast <ip
   address> ] [ unit <id> ] [ tag <vlan tag> ]
```

Description

Configures shared IP addresses on installed network interface cards. The configuration you create with this command is stored in the BIG/db. If you use VLAN tags in your configuration, you can use this command to set the VLAN tag for the shared IP alias.

You must issue this command for each interface that you want configure with the same IP alias. For example, if you want to configure the IP alias 192.168.100.100 for the interfaces **exp0** and **exp1**, type the following comands:

```
bigpipe ipalias exp0 192.168.100.100 netmask 255.255.0.0
bigpipe ipalias exp1 192.168.100.100 netmask 255.255.0.0
```

-l

bigpipe -l <file_name>

Description

Use the **-l** command to load the BIG-IP Controller configuration from <**file_name**> without resetting the current configuration.

lb

```
bigpipe lb show
bigpipe lb round_robin | rr
bigpipe lb ratio
bigpipe lb priority
bigpipe lb fastest
bigpipe lb least_conn
bigpipe lb predictive
bigpipe lb observed
```

Description

Sets the global load balancing mode for all node list virtual servers.

Note

Pools are configured with their own load balancing method. For more information on this, see Setting up persistence for a pool, on page 1-65.

Setting the load balancing mode

Use the following syntax to set the load balancing mode:

```
bigpipe lb <mode name>
```

The mode names are displayed at the top of the page.

This command sets the load balancing mode to Least Connections, which routes new connections to the node that currently maintains the least number of connections.

```
bigpipe lb least_conn
```

Viewing the currently selected load balancing mode

The following command displays the currently selected load balancing mode.

```
bigpipe lb show
```

maint

bigpipe maint

Description

Toggles a BIG-IP Controller into and out of Maintenance mode. When in Maintenance mode, a BIG-IP Controller accepts no new connections, but it does allow existing connections to complete.

The **maint** command interactively prompts you to enter or exit the maintenance mode.

```
bigpipe maint
```

If the BIG-IP Controller is already in maintenance mode, the **maint** command takes the BIG-IP Controller out of maintenance mode. If the BIG-IP Controller is in maintenance mode for more than 20 minutes, that the BIG-IP Controller immediately begins to accept new connection requests.

If the BIG-IP Controller has been in maintenance mode for more than 20 minutes, that it automatically updates all network ARP caches; this process normally takes a few seconds. However, you can speed the process up by reloading the configuration file, using the following command:

```
bigpipe -f /etc/bigip.conf
```

mirror

bigpipe mirror enable | disable | show

Description

Enables and disables mirroring between active and standby BIG-IP Controllers. Mirroring ensures that persistence and connection information on the active controller are duplicated on the standby controllers. This command enables and disables mirroring for all virtual servers.

To enable mirroring on a redundant system:

bigpipe mirror enable

To disable mirroring on a redundant system:

bigpipe mirror disable

To show the current status of mirroring on a redundant system:

bigpipe mirror show

nat

```
bigpipe nat <orig_addr> to <trans_addr>[/<bitmask>] [<ifname>]
   [unit <unit ID>]
bigpipe nat <orig_addr> to <trans_addr> netmask <netmask> \
   [broadcast <broadcast_ip>] [<ifname>] [unit <unit ID>]
bigpipe nat <orig_addr> [...<orig_addr>] delete
bigpipe nat <trans_addr> [...<trans_addr>] delete
bigpipe nat [<trans_addr> [...<trans_addr>] ] show
bigpipe nat [<orig_addr> [...<orig_addr>] ] show
bigpipe nat [<orig_addr>] stats reset
```

Description

Defines an IP address, routable on the external network, that a node can use to initiate connections to hosts on the external network and receive direct connections from clients on the external network. The NAT (Network Address Translation) command defines a mapping between the IP address of a server behind the BIG-IP Controller <**orig_addr**> and an unused routable address on the network in front of the BIG-IP Controller <**trans_addr**>.

Defining a NAT

A NAT definition maps the IP address of a node <**orig_addr**> to a routable address on the external interface <**trans_addr**>, and can include an optional interface and netmask specification. Use the following syntax to define a NAT:

```
bigpipe nat <orig_addr> to <trans_addr>[/<bitmask>] [<ifname>]
   [unit <unit ID>]
```

The <**ifname**> parameter is the internal interface of the BIG-IP Controller through which packets must pass to get to the destination internal address. The BIG-IP Controller can determine the interface to configure for the NAT in most cases. The <**ifname**> parameter is useful, for example, where there is more than one

internal interface. You can use the **unit <unit ID>** parameter to specify the controller to which this NAT applies in an active-active redundant system.

The following example shows a NAT definition:

```
bigpipe nat 10.10.10.10 to 10.12.10.10/24 exp1
```

Deleting NATs

Use the following syntax to delete one or more NATs from the system:

```
bigpipe nat <orig_addr> [...<orig_addr>] delete
```

Displaying status of NATs

Use the following command to display the status of all NATs included in the configuration:

```
bigpipe nat show
```

See figure 2.3 for the output when you display the status of a NAT. Use the following syntax to display the status of one or more selected NATs:

```
bigpipe nat <orig_addr> [...<orig_addr>] show
```

```
NAT { 10.10.10.3 to 9.9.9.9 }
 (pckts,bits) in = (0, 0), out = (0, 0)
NAT { 10.10.10.4 to 12.12.12.12
netmask 255.255.255.0 broadcast 12.12.12.255 }
 (pckts,bits) in = (0, 0), out = (0, 0)
```

Figure 2.3 *Output when you display the status of a NAT*

Resetting statistics for a NAT

Use this command to reset the statistics for an individual NAT:

```
bigpipe nat [<orig_addr>] stats reset
```

Use the following command to reset the statistics for all NATs:

```
bigpipe nat stats reset
```

Additional Restrictions

The **nat** command has the following additional restrictions:

- The IP address defined in the **<orig_addr>** parameter must be routable to a specific server behind the BIG-IP Controller.
- You must delete a NAT before you can redefine it.
- The interface for a NAT may only be configured when the NAT is first defined.

-n

bigpipe -n

Description

Use the **-n** option in combination with other commands, such as **bigpipe vip**, to display ports numerically rather than by service name. For example, type the following command to display ports numerically:

bigpipe -n vip

Notice that the ports are listed numerically rather than by service name. See Figure 2.4 for an example of the output.

```
VIP +------> 11.100.1.1          UNIT 1
    |            (cur, max, limit, tot) = (0, 0, 0, 0)
    |            (pckts,bits) in = (0, 0), out = (0, 0)
    +---+--> PORT 80                    UP
        |        (cur, max, limit, tot) = (0, 0, 0, 0)
        |        (pckts,bits) in = (0, 0), out = (0, 0)
        MEMBER 11.12.1.100:80           UP
                 (cur, max, limit, tot) = (0, 0, 0, 0)
                 (pckts,bits) in = (0, 0), out = (0, 0)
```

Figure 2.4 *The output of* ***bigpipe -n vip***

node

```
bigpipe node <node ip>[:<port>][...<node ip>[:<port>]] \
    enable | disable
bigpipe node [<node ip>[:<port>][...<node ip>[:<port>]] ] show
bigpipe node <node ip>[:<port>][...<node ip>[:<port>]] \
    limit <max conn>
bigpipe node <node ip>[:port] up | down
bigpipe node [<node ip>:<port>] stats reset
```

Description

Displays information about nodes and allows you to set properties for nodes, and node addresses.

Enabling and disabling nodes and node addresses

To enable a node address, use the **node** command with a node address and the **enable** option:

```
bigpipe node 192.168.21.1 enable
```

To disable a node address, use the **node** command with the **disable** option:

```
bigpipe node 192.168.21.1 disable
```

To enable one or more node addresses, use the **node** command with a node address and port, and the **enable** option:

```
bigpipe node 192.168.21.1:80 enable
```

To disable one or more node addresses, use the **node** command with the **disable** option:

```
bigpipe node 192.168.21.1:80 disable
```

Marking nodes and node ports up or down

To mark a node address down, use the **node** command with a node address and the **down** option. (Note that marking a node down prevents the node from accepting new connections. Existing connections are allowed to complete.):

```
bigpipe node 192.168.21.1 down
```

To mark a node address up, use the **node** command with the **up** option:

```
bigpipe node 192.168.21.1 up
```

To mark a port down, use the **node** command with a node address and port, and the **down** option. (Note that marking a port down prevents the port from accepting new connections. Existing connections are allowed to complete.):

```
bigpipe node 192.168.21.1:80 down
```

To mark a port up, use the **node** command with **up** option:

```
bigpipe node 192.168.21.1:80 up
```

Setting connection limits for nodes

Use the following command to set the maximum number of concurrent connections allowed on a node:

```
bigpipe node <node ip>[:<port>][...<node ip>[:<port>]] \
    limit <max conn>
```

Note that to remove a connection limit, you also issue the preceding command, but set the <**max conn**> variable to **0** (zero). For example:

```
bigpipe node 192.168.21.1:80 limit 0
```

Setting connection limits for node addresses

The following example shows how to set the maximum number of concurrent connections to **100** for a list of node addresses:

```
bigpipe node 192.168.21.1 192.168.21.1
    192.168.21.1 limit 100
```

To remove a connection limit, you also issue this command, but set the <**max conn**> variable to **0** (zero).

Displaying status of all nodes

`bigpipe node show`

When you issue the **node show** command, the BIG-IP Controller displays the node status (**up** or **down**, or **unchecked**), and a node summary of connection statistics, which is further broken down to show statistics by port. The report shows the following information:

- current number of connections
- total number of connections made to the node since last boot
- maximum number of concurrent connections since the last boot
- concurrent connection limit on the node
- total number of connections made to the node since last boot
- total number of inbound and outbound packets and bits

Figure 2.5 shows the output of this command.

```
bigpipe node 192.168.200.50:20
NODE 192.168.200.50    UP
|    (cur, max, limit, tot) = (0, 0, 0, 0)
|    (pckts,bits) in = (0, 0), out = (0, 0)
+-   PORT 20           UP
     (cur, max, limit, tot) = (0, 0, 0, 0)
     (pckts,bits) in = (0, 0), out = (0, 0)
```

Figure 2.5 *Node status and statistics*

Displaying the status of individual nodes and node addresses

Use this command to display status and statistical information for one or more node addresses:

```
bigpipe node 192.168.21.1 show
```

The command reads the status of each node address, the number of current connections, total connections, and connections allowed, and the number of cumulative packets and bits sent and received.

Use the following command to display status and statistical information for one or more specific nodes:

```
bigpipe node 192.168.21.1:80 show
```

Resetting statistics for a node

Use the following command to reset the statistics for an individual node address:

```
bigpipe node [<node ip>:<port>] stats reset
```

persist

bigpipe persist <port> [...<port>] <seconds>

bigpipe persist dump

Description

Enables or disables simple persistence on one or more virtual ports. Persistence tracks the source IP addresses of all incoming requests, and the nodes and ports that hosted the request. It forces new connections from the source address to use the same node as used by the prior connection from that source IP address. A configurable time limit determines how long the BIG-IP Controller retains persistent connection information. By default, persistence is disabled on all ports. Persistence is affected by certain system control variables.

Setting a persistence timeout

Use the following syntax to set the number of seconds for which the BIG-IP Controller maintains persistent connection information on a specific virtual port:

bigpipe persist <port> <seconds>

Set <**seconds**> to **0** to turn persistence **off** for a specific virtual port.

Displaying persistent connections

Use the following syntax to display information about current persistent connections:

bigpipe persist [<port>] [...port] dump

pool

```
bigpipe pool <pool name> { lb_method <lb_method_specification>
   [persist_mode <persist_mode_specification>] <member
   definition>... }
```

```
bigpipe pool <pool name> add { <member definition>... }
```

```
bigpipe pool <pool name> delete { <member definition>... }
```

```
bigpipe pool <pool name> modify { [lb_method
   <lb_method_specification>] [persist_mode
   <persist_mode_specification>] <member  definition>... }
```

```
bigpipe pool <pool name> delete
```

```
bigpipe pool [<pool name>] show
```

```
bigpipe pool <pool name> lb_method show
```

```
bigpipe pool <pool name> persist show
```

Description

Use the **pool** command to create, delete, modify, or display the pool definitions on the BIG-IP Controller. Use pools to group members together with a common load balancing mode and persistence mode. For additional information about configuring pools, see the ***BIG-IP Controller Administrator Guide***, *Working with Intelligent Traffic Control*.

Creating a pool

To create a pool use the following syntax:

```
bigpipe pool <pool_name> {lb_method <lb_method_specification>
   [persist_mode <persist_mode_specification>]
   <member_definition>... member <member_definition>}
```

Each of these elements is described in Table 2.5, on page 2-42.

For detailed information about setting up persistence with pools, see the ***Administrator Guide***, *Setting up persistence for a pool*, on page 1-65.

To activate Insert HTTP cookie persistence from the command line

If you specify Insert mode, the BIG-IP Controller inserts a cookie from the server in the header of the HTTP response with information about the server to which the client connects. The cookie is named **BIGipServer<pool_name>**, and it includes the address and port of the server handling the connection. The expiration date for the cookie is set based on the timeout configured on the BIG-IP Controller.

To activate Insert mode from the command line, use the following syntax:

```
bigpipe pool <pool_name> { <lb_method_specification> persist_mode
   cookie cookie_mode insert cookie_expiration <timeout> <member
   definition> }
```

The **<timeout>** value for the cookie is written using the following format:

```
<days>d hh:mm:ss
```

To activate Rewrite mode cookie persistence from the command line

If you specify Rewrite mode, the BIG-IP Controller intercepts a Set-Cookie, named **BIGipCookie**, sent from the server to the client and overwrites the name and value of the cookie. The new cookie is named **BIGipServer<pool_name>** and it includes the address and port of the server handling the connection.

To use rewrite mode, you must set up the cookie created by the server. For Rewrite mode to work, the BIG-IP Controller needs a blank cookie coming from the web server to rewrite. With Apache variants, you can add the cookie to every web page header by adding an entry in the **httpd.conf** file:

```
Header add Set-Cookie BIGipCookie=
   0000000000000000000000000000000000000000...
```

The cookie should contain a total of 120 zeros.

WARNING

For backward compatibility the blank cookie can contain only 75 zeros. However, cookies of this size do not allow you to use rules and persistence together.

To activate Rewrite mode from the command line, use the following syntax:

```
bigpipe pool <pool_name> { <lb_method_specification> persist_mode
   cookie cookie_mode rewrite cookie_expiration <timeout> <member
   definition> }
```

The **<timeout>** value for the cookie is written using the following format:

```
<days>d hh:mm:ss
```

To activate Passive mode cookie persistence from the command line

If you specify Passive mode, the BIG-IP Controller does not insert or search for blank **Set-Cookies** in the response from the server. It does not try to set up the cookie. In this mode, BIG-IP Controller assumes that the server provides the cookie formatted with the correct node information and timeout.

In order for Passive mode to work, a cookie needs to come from the web server with the appropriate node information in the cookie. With Apache variants, you can add the cookie to every web page header by adding an entry in the **httpd.conf** file:

```
Header add Set-Cookie: "BIGipServerMY_POOL=184658624.20480.000;
   expires=Sat, 19-Aug-2000 19:35:45 GMT; path=/"
```

In this example, **my_pool** is the name of the pool that contains the server node, **184658624** is the encoded node address and **20480** is the encoded port.

The equation for an address (a.b.c.d) is:

d*256^3 + c*256^2 + b*256 +a

The way to encode the port is to take the two bytes that store the port and reverse them. So, port 80 becomes 80 * 256 + 0 = 20480. Port 1433 (instead of 5 * 256 + 153) becomes 153 * 256 + 5 = 39173.

After you set up the cookie created by the web server, you must activate Passive mode on the BIG-IP Controller. To activate HTTP cookie persistence from the command line, use the following syntax:

```
bigpipe pool <pool_name> { <lb_method_specification> persist_mode
   cookie cookie_mode passive <member definition> }
```

Note

The ***<timeout>*** *value is not used in Passive mode.*

To configure the hash cookie persistence option from the command line

If you specify hash mode, the hash mode consistently maps a cookie value to a specific node. When the client returns to the site, the BIG-IP Controller uses the cookie information to return the client to a given node. With this mode, the web server must generate the cookie. The BIG-IP Controller does not create the cookie automatically like it does with insert mode.

Use the following syntax to configure the hash cookie persistence option:

```
bigpipe pool <pool_name> { <lb_method_specification> persist_mode
   cookie cookie_mode hash cookie_hash_name <cookie_name>
   cookie_hash_offset <cookie_value_offset> cookie_hash_length
   <cookie_value_length> <member definition> }
```

The **<cookie_name>**, **<cookie_value_offset>**, and **<cookie_value_length>** values are described in Table 2.4:

Hash mode values	Description
`<cookie_name>`	This is the name of an HTTP cookie being set by a Web site.
`<cookie_value_offset>`	This is the number of bytes in the cookie to skip before calculating the hash value.
`<cookie_value_length>`	This is the number of bytes to use when calculating the hash value.

Table 2.4 *The cookie hash mode values*

To activate sticky persistence from the command line

Use the following command to enable sticky persistence for a pool:

```
bigpipe pool <pool_name> modify { persist_mode sticky <enable |
   disable> sticky_mask <ip address> }
```

Use the following command to disable sticky persistence for a pool:

```
bigpipe pool <pool_name> modify { persist_mode sticky disable
   sticky_mask <ip address> }
```

Use the following command to delete sticky entries for the specified pool:

```
bigpipe pool <pool_name> sticky clear
```

To activate SSL persistence from the command line

Use the following syntax to activate SSL persistence from the command line:

```
bigpipe pool <pool_name> modify { persist_mode ssl ssl_timeout
   <timeout> simple_mask <ip_mask> }
```

For example, if you want to set SSL persistence on the pool **my_pool**, type the following command:

```
bigpipe pool my_pool modify { persist_mode ssl ssl_timeout 3600
   simple_mask 255.255.255.0 }
```

To apply a simple timeout and persist mask from the command line

The complete syntax for the command is:

```
bigpipe pool <pool_name> modify { [<lb_method_specification>]
   persist_mode simple simple_timeout <timeout> simple_mask
   <dot_notation_longword> }
```

For example, the following command would keep persistence information together for all clients within a C class network that connect to the pool **classc_pool**:

```
bigpipe pool classc_pool modify { persist_mode simple
   simple_timeout 1200 simple_mask 255.255.255.0 }
```

You can turn off a persist mask on a pool by using the **none** option in place of the **simple_mask** mask. To turn off the persist mask that you set in the preceding example, use the following command:

```
bigpipe pool classc_pool modify { simple_mask none }
```

Display persistence information for a pool

To show the persistence configuration for the pool:

```
bigpipe pool <pool_name> persist show
```

To display all persistence information for the pool named **classc_pool**, use the **show** option:

```
bigpipe pool classc_pool persist show
```

Options

Use the following elements to construct pools:

Pool Element	Description
Pool name	A string from 1 to 31 characters, for example: **new_pool**
Member definition	member <ip address>:<port> [ratio <value>] [priority <value>]
lb_method_specificaton	lb_method [rr | ratio | priority | fastest | least_conn | predictive | observed | ratio_member | priority_member | least_conn_member]
persist_mode_specification	persist_mode [cookie | simple | ssl | sticky]

Table 2.5 *The elements you can use to construct a pool*

Deleting a pool

To delete a pool, use the following syntax:

```
bigpipe pool <pool_name> delete
```

All references to a pool must be removed before a pool can be deleted.

Modifying pools

You can use the command line to add or delete members from a pool. You can also modify the load balancing mode for a pool from the command line. To add a new member to a pool use the following syntax:

```
bigpipe pool <pool_name> add { 1.2.3.2:telnet }
```

To delete a member from a pool use the following syntax:

```
bigpipe pool <pool_name> delete { 1.2.3.2:telnet }
```

Display pools

Use the following syntax to display all pools:

```
bigpipe pool show
```

Use the following syntax to display a specific pool:

```
bigpipe pool <pool_name> show
```

port

```
bigpipe port <port> [...<port>] limit <max conn>
bigpipe port <port> [...<port>] enable | disable | show
```

Description

Enables and disables network traffic on virtual ports, and also sets connection limits on ports. You can use standard port numbers, service or port names (for example, **www**, **http**, or **80**) for the <**port**> parameter. Note that the port settings you define with this command control the port service for all virtual servers that use the port. By default, all ports are disabled.

A port is any valid port number, between 0 and 65535, inclusive, or any valid service name in the **/etc/services** file.

Allowing and denying virtual ports

You can enable or disable traffic to specific virtual ports. The default setting for all virtual ports is disabled. Use the following syntax to allow one or more virtual ports:

```
bigpipe port <port> [...<port>] enable
```

To deny access to one or more virtual ports, use this syntax:

```
bigpipe port <port> [...<port>] disable
```

Setting connection limits on ports

Use the following syntax to set the maximum number of concurrent connections allowed on a virtual port. Note that you can configure this setting for one or more virtual ports.

```
bigpipe port <port> [...<port>] limit <max conn>
```

To turn off a connection limit for one or more ports, use the same command, setting the <**max conn**> parameter to **0** (zero) like this:

```
bigpipe port <port> [...<port>] limit 0
```

Displaying the status of all virtual ports

Use the following syntax to display the status of virtual ports included in the configuration:

bigpipe port show

Displaying the status for specific virtual ports

Use the following syntax to display the status of one or more virtual ports:

bigpipe port <port> [...<port>] show

Figure 2.6 shows a sample of formatted output from the **port** command.

```
bigpipe port telnet show

PORT  23     telnet       enable
(cur, max, limit, tot, reaped) = (37,73,100,691,29)
      (pckts,bits) in = (2541, 2515600), out = (2331, 2731687)
```

Figure 2.6 *Formatted output of* ***port*** *command showing the Telnet port statistics*

proxy

```
bigpipe proxy <ip>:<port> [/bitmask] [<ifname>] [<unit id>] target
   <server | vip> <ip>:<port> ssl enable key <key> cert <cert>
```

```
bigpipe proxy <ip>:<port> [<ifname>] [<unit id>] netmask <ip>
   [broadcast <ip>]  target <server | vip> <ip>:<port> ssl enable
   key <key> cert <cert>
```

```
bigpipe proxy <ip>:<port> enable
```

```
bigpipe proxy <ip>:<port> disable
```

```
bigpipe proxy <ip>:<port> delete
```

```
bigpipe proxy <ip>:<port> show
```

```
bigpipe proxy <ip>:<port> lasthop pool <pool_name>
```

Description

Use the proxy command to create, delete, modify, or display the SSL gateway definitions on the BIG-IP Controller. For detailed information about setting up the SSL Accelerator feature, see the ***BIG-IP Administrator Guide***, *Configuring an SSL Accelerator*.

Creating an SSL gateway from the command line

Use the following command syntax to create an SSL gateway. Use this syntax if you want to configure a gateway by specifying a bitmask instead of a netmask and broadcast address:

```
bigpipe proxy <ip>:<port> [/bitmask] [<ifname>] [<unit id>] target
   <server | vip> <ip>:<port> ssl enable key <key> cert <cert>
```

Use this syntax if you want to configure a gateway by specifying a netmask and broadcast address instead of a bitmask:

```
bigpipe proxy <ip>:<port> [<ifname>] [<unit id>] netmask <ip>
   [broadcast <ip>]  target <server | vip> <ip>:<port> ssl enable
   key <key> cert <cert>
```

For example, from the command line you can create an SSL gateway that looks like this:

```
bigpipe proxy 10.1.1.1:443 exp0 unit 1 { netmask 255.255.255.0
   broadcast 10.1.1.255 target vip 20.1.1.1:80 ssl enable key
   my.server.net.key cert my.server.net.cert }
```

Note that when the configuration is written out in the **bigip.conf** file, the line **ssl enable** is automatically added. When the SSL gateway is written in the **/etc/bigip.conf** file, it looks like the sample in Figure 2.7:

```
proxy 10.1.1.1:443 exp0 unit 1 {
    netmask 255.255.255.0
    broadcast 10.1.1.255
    target vip 20.1.1.1:80
    ssl enable
    key my.server.net.key
    cert my.server.net.cert
}
```

Figure 2.7 *An example SSL gateway configuration*

Enabling, disabling, or deleting an SSL gateway from the command line

You can enable, disable, or delete an SSL gateway with the following syntax:

```
bigpipe proxy <ip>:<port> enable
bigpipe proxy <ip>:<port> disable
bigpipe proxy <ip>:<port> delete
```

If you want to enable the SSL gateway 209.100.19.22:443, you might type the following command:

```
bigpipe proxy 209.100.19.22:443 enable
```

If you want to disable the SSL gateway 209.100.19.22:443, you could type the following command:

```
bigpipe proxy 209.100.19.22:443 disable
```

For example, if you want to delete the SSL gateway 209.100.19.22:443, type the following command:

bigpipe proxy 209.100.19.22:443 delete

Displaying configuration information for an SSL gateway from the command line

Use the following syntax to view the configuration for the specified SSL gateway:

bigpipe proxy <ip>:<port> show

For example, if you want to view configuration information for the SSL gateway 209.100.19.22:80, type the following command:

bigpipe proxy 209.100.19.22:80 show

Figure 2.8 is a sample output from the **bigpipe proxy show** command.

```
SSL PROXY +---> 11.12.1.200:443 -- Originating Address -- Enabled   Unit 1
          |     Key File Name balvenie.scotch.net.key
          |     Cert File Name balvenie.scotch.net.cert
          |     LastHop Pool Name
          +===> 11.12.1.100:80 -- Destination Address -- Server


SSL PROXY +---> 11.12.1.120:443 -- Originating Address -- Enabled   Unit 1
          |     Key File Name balvenie.scotch.net.key
          |     Cert File Name balvenie.scotch.net.cert
          |     LastHop Pool Name
          +===> 11.12.1.111:80 -- Destination Address -- Vip
```

Figure 2.8 *Output from the* ***bigpipe proxy show*** *command*

Adding a last hop pool to an SSL gateway from the command line

Use the following syntax to reference a last hop pool from an SSL gateway:

bigpipe proxy <ip>:<port> lasthop pool <pool_name>

For example, if you want to assign the last hop pool named **ssllasthop_pool** to the SSL gateway **11.12.1.200:443**, type the following command:

```
bigpipe proxy 11.12.1.200:443 lasthop pool
   ssllasthop_pool
```

-r

bigpipe -r

Description

Use the following syntax to clear the configuration values and counter values from memory:

bigpipe -r

WARNING

Use this command with caution. All network traffic stops when you run this command.

Typically, this command is used on a standby BIG-IP Controller prior to loading a new **/etc/bigip.conf** file that contains new **tping** and **treaper** values.

For example, you can execute the following commands on a standby BIG-IP Controller:

bigpipe -r

bigpipe -f <filename>

This sequence of commands ensures that only the values set in the **<filename>** specified are in use.

ratio

```
bigpipe ratio [<node ip>] [node ip> ...] show
bigpipe ratio <node ip> [<node ip>...] <weight>
```

Description

This command provides two functions related to load balancing:

- For the Ratio load balancing mode, the command sets the weight or proportions for one or more node addresses.
- For the Priority load balancing mode, the command sets the priority level. Note that multiple node addresses can have the same priority level setting.

Setting ratio weight for one or more node addresses

The default ratio setting for any node address is **1**. If you use the Ratio or Priority load balancing modes, you must set a ratio other than **1** for at least one node address in the configuration. If you do not change at least one ratio setting, the load balancing modes have the same affect as the Round Robin load balancing mode.

Use the following syntax to set the ratio for one or more node addresses:

```
bigpipe ratio <node ip> [...<node ip>] <weight>
```

For example, the following command sets the ratio weight to **3** for a specific node address:

```
bigpipe ratio 192.168.103.20 3
```

Displaying the ratio weights for node addresses

The following command displays the current ratio weight settings for all node addresses.

```
bigpipe ratio show
```

The command displays this output:

```
192.168.200.51    ratio = 3
```

```
192.168.200.52    ratio = 1
```

Displaying ratio weight for specific node addresses

Use the following syntax to display the ratio setting for one or more node addresses:

`bigpipe ratio <node ip> [...<node ip>] show`

Note

*The <**weight**> parameter must be a whole number, equal to or greater than **1**.*

rule

```
bigpipe rule <rule name> ' { <if statement> | <use statement> } '
bigpipe rule <rule name> delete
bigpipe rule [<rule name>] show
```

Description

Use the **rule** command to create, delete, or display the rules on the BIG-IP Controller. Rules allow a virtual server to access any number of pools on the BIG-IP Controller. For more detailed information about using rules, see *Rule*, on page 1-114.

◆ Note

Before you define a rule, you must define the pool or pools that you want the rule to reference.

Create a rule

You can add rules by manually typing them into an existing **/etc/bigip.conf** file. However, you can also use the **bigpipe rule** command to create, delete, or display rules. To create a rule with **bigpipe**, type the complete rule on the command line, without line breaks. For example, you can type in this rule:

```
bigpipe rule cgi_rule ' {if (http_uri ends_with "cgi") {use (
   cgi_pool )} else {use ( default_pool )}} '
```

If the **http_uri** string ends with **"cgi"** then the members from **cgi_pool** are used for load balancing. If the **http_uri** string does not end with **"cgi"**, then the members of **default_pool** are used for load balancing.

Associating a rule with virtual server

You can associate a rule with a virtual server by using the following syntax:

```
bigpipe vip <virt ip>:<port> use rule <rule_name>
```

For example, if you want to associate the rule **cgi_rule** to the virtual server **10.20.2.101:http**, type in the following command:

```
bigpipe vip 10.20.2.101:http use rule cgi_rule
```

Deleting a rule

You can delete a rule using the following syntax:

```
bigpipe rule <rule_name> delete
```

Display rules

Use the following syntax to display all rules:

```
bigpipe rule show
```

Use the following syntax to display a specific rule:

```
bigpipe rule <rule_name> show
```

Definitions

You can create a rule by combining a number of different elements. A simple rule could contain the following elements:

```
rule <rule_name> { if ( <variable> <binary_operator> "<literal>" )
   { use ( <pool_name> ) } else { use ( <another_pool_name> ) } }
```

For example, a rule named **cgi_rule** that sends CGI connections to a load balancing pool named **cgi_pool**, or HTTP connections to a pool named **http_pool** looks like this:

```
bigpipe rule cgi_rule ' {if (http_uri ends_with "cgi") {use (
   cgi_pool )} else {use ( http_pool )}} '
```

Use the elements in Table 2.6 to create rules.

Element	Description
A rule definition is	`rule { <if statement> }`

A statement is	`<if_statement>`
	`<use_statement>`
	`discard`

A **use** statement	`use ( <pool_name> )`
An **if** statement	`if ( <expression> ) { <statement> }` `[ { else <statement> } ]`

An expression	`<literal>`
	`<variable>`
	`( <expression> )`
	`exist <variable>`
	`not <expression>`
	`<expression> <binary_operator> <expression>`

literal	`<regex_literal>`
	`<string_literal>`
	`<address_literal>`

A regular expression literal	Is a string of 1 to 63 characters enclosed in quotes that may contain regular expressions
A string literal	Is a string of 1 to 63 characters enclosed in quotes
An address literal	`<dot_notation_longword> [netmask <dot_notation_longword>]`

Table 2.6 *The elements you can use to construct rules*

Element	Description
Dot notation longword	**<0-255>.<0-255>.<0-255>.<0-255>**
variable	**http_method** **http_header** **http_version** **http_uri** **http_host** **http_cookie <cookie_name>** **client_addr** **ip_protocol**
binary operator	**or** **and** **contains** **matches** **equals** **starts_with** **ends_with** **matches_regex**

Table 2.6 *The elements you can use to construct rules*

Note

For more detailed information about using rules, see Rule, on page 1-114.

-s

```
bigpipe -s [ <filename> | - ]
```

Description

Writes the current BIG-IP Controller configuration settings from memory to the default boot configuration file named **/etc/bigip.conf**.

You can type **bigpipe -s**, or a hyphen character (**-**) in place of a file name, to display the configuration on the standard output device.

```
bigpipe -s -
```

Or you can type the following command:

```
bigpipe -s
```

If you are testing and integrating BIG-IP Controllers into a network, you may want to use multiple test configuration files. Use the following syntax to write the current configuration to a file name that you specify:

```
bigpipe -s <filename>
```

For example, the following command saves the current configuration from memory to an alternate configuration file named **/etc/bigip.conf2**.

```
bigpipe -s /etc/bigip.conf2
```

snat

```
bigpipe snat map <node ip> [...<node ip>] to \
   <SNAT ip> [netmask <ip>] [<ifname>] [unit <unit ID>]
bigpipe snat map default to <SNAT ip> [<ifname>] \
   [unit <unit ID>] [netmask <ip>]
bigpipe snat <SNAT ip> [...<SNAT ip>] delete
bigip snat default delete
bigpipe snat default dump [verbose]
bigpipe snat [<node ip> [...<node ip>] ] dump [verbose]
bigpipe snat globals show
bigpipe snat default show
bigpipe snat [<node ip> [...<node ip>] ] show
bigpipe snat limit <max conn>
bigpipe snat default limit <max conn>
bigpipe snat <node ip> [...<node ip>] limit \
   <max conn>
bigpipe snat <node ip> [...<node ip>] mirror \
   enable | disable
bigpipe snat default mirror enable | disable
bigpipe snat <node ip> [...<node ip>] timeout tcp | udp \
   <seconds>
bigpipe snat [default] timeout tcp | udp <seconds>
bigpipe snat <SNAT ip> [...<SNAT ip>] stats reset
bigpipe snat default stats reset
```

Description

Defines one or more addresses that nodes can use as a source IP address when initiating connections to hosts on the external network. Note that clients cannot use SNAT addresses to connect directly to nodes.

Defining the default SNAT

Use the following syntax to define the default SNAT. If you use the netmask parameter and it is different from the external interface default netmask, the command sets the netmask and derives the broadcast address. You can use the **unit** <**unit ID**> parameter to specify a unit in an active-active redundant configuration.

```
bigpipe snat map default to <SNAT ip> [<ifname>] [unit <unit ID>]
   [netmask <ip>]
```

Creating individual SNAT addresses

Use the following command syntax to create a SNAT mapping:

```
bigpipe snat map <node ip> [...<node ip>] to \
   <SNAT ip> [<ifname>] [unit <unit ID>] [netmask <ip>]
```

If the netmask is different from the external interface default netmask, the command sets the netmask and derives the broadcast address.

Deleting SNAT Addresses

The following syntax deletes a specific SNAT:

```
bigpipe snat <SNAT ip> | default delete
```

Showing SNAT mappings

The following **bigpipe** command shows mappings:

```
bigpipe snat [<SNAT ip>] [...<SNAT ip>] show
bigpipe snat default show
```

The following command shows the current SNAT connections:

```
bigpipe snat [<SNAT ip>] [...<SNAT ip>] dump [ verbose ]
bigpipe snat default dump [ verbose ]
```

The optional **verbose** keyword provides more detailed output.

The following command prints the global SNAT settings:

```
bigpipe snat globals show
```

Limiting connections

Use the following commands to set the maximum number of concurrent connections allowed for one or more SNAT addresses. Zero indicates no limit.

```
bigpipe snat <SNAT ip> limit <max conn>
```

The default SNAT address connection limit is set with the following command:

```
bigpipe snat default limit <max conn>
```

Set the global concurrent connection limit with this command:

```
bigpipe snat limit <max conn>
```

Enabling mirroring for redundant systems

The following example sets SNAT mirroring for all SNAT connections originating at 192.168.225.100:

```
bigpipe snat 192.168.225.100 mirror enable
```

Setting idle connection timeouts

Use the following command to set the timeout for idle TCP connections:

```
bigpipe snat timeout tcp <seconds>
```

Use the following command to set the timeout for idle UDP connections. Note that you must have a timeout set for UDP connections; zero is not allowed:

```
bigpipe snat timeout udp <seconds>
```

Use the following command to set the timeout for idle TCP connections originating at this node address. Set <**seconds**> to **0** (zero) to disable TCP timeout for these nodes.

```
bigpipe snat <node ip> [...<node ip>] timeout tcp <seconds>
```

Use the following command to set the timeout for idle TCP connections originating at the default node address. Set <**seconds**> to **0** (zero) to disable TCP timeout for these nodes.

```
bigpipe snat default timeout tcp <seconds>
```

Use the following syntax to set the timeout for idle UDP connections originating at this node address. Note that you must have a timeout set for UDP connections; zero is not allowed:

```
bigpipe snat <node ip> [...<node ip>] timeout udp <seconds>
```

Use the following syntax to set the timeout for idle UDP connections originating at the default SNAT address. Note that you must have a timeout set for UDP connections; zero is not allowed:

```
bigpipe snat default timeout udp <seconds>
```

Clearing statistics

You can reset statistics by node or by SNAT address. Use the following syntax to clear all statistics for one or more nodes:

```
bigpipe snat <node ip> [ ...<node ip> ] stats reset
```

Use the following syntax to clear all statistics for one or more SNAT addresses:

```
bigpipe snat <SNAT ip> [ ...<SNAT ip> ] stats reset
```

Use the following command to reset the statistics to zero for the default:

```
bigpipe snat default stats reset
```

summary

bigpipe summary

Description

Displays a summary of current usage statistics.

The output display format for the **summary** command is shown in Figure 2.9.

For detailed descriptions of each of statistic displayed by the **summary** command, refer to the ***BIG-IP Controller Administrator Guide***, *Monitoring and Administration*.

```
BIG-IP total uptime           = 1 (day) 4 (hr) 40 (min) 8 (sec)
   BIG-IP total uptime (secs)    = 103208
   BIG-IP total # connections    = 0
   BIG-IP total # pkts           = 0
   BIG-IP total # bits           = 0
   BIG-IP total # pkts(inbound)  = 0
   BIG-IP total # bits(inbound)  = 0
   BIG-IP total # pkts(outbound) = 0
   BIG-IP total # bits(outbound) = 0
   BIG-IP error no nodes available          = 0
   BIG-IP tcp port deny                     = 0
   BIG-IP udp port deny                     = 0
   BIG-IP vip tcp port deny                = 0
   BIG-IP vip udp port deny                 = 0
   BIG-IP max connections deny              = 0
   BIG-IP vip duplicate syn ssl             = 0
   BIG-IP vip duplicate syn wrong dest      = 0
   BIG-IP vip duplicate syn node down       = 0
   BIG-IP vip maint mode deny               = 0
   BIG-IP virtual addr max connections deny = 0
   BIG-IP virtual path max connections deny = 0
   BIG-IP vip non syn                       = 0
   BIG-IP error not in out table            = 0
   BIG-IP error not in in table             = 0
   BIG-IP error vip fragment no port        = 0
   BIG-IP error vip fragment no conn        = 0
   BIG-IP error standby shared drop         = 0
   BIG-IP dropped inbound                   = 0
   BIG-IP dropped outbound                  = 0
   BIG-IP reaped                            = 0
   BIG-IP ssl reaped                        = 0
   BIG-IP persist reaped                    = 0
   BIG-IP udp reaped                        = 0
   BIG-IP malloc errors                     = 0
   BIG-IP bad type                          = 0
   BIG-IP mem pool total 96636758 mem pool used 95552 mem percent
   used   0.10
```

Figure 2.9 *The* ***summary*** *output display*

timeout_node

```
bigpipe timeout_node show
bigpipe timeout_node <seconds>
bigpipe timeout_node 0
```

Description

Sets the amount of time that a server has to respond to a BIG-IP Controller ping in order for the server to be marked **up**. If a server fails to respond within the specified time, the BIG-IP Controller assumes that the server is down, and the BIG-IP Controller no longer sends packets to the services hosted by the server. If the server responds to the next ping, or to subsequent pings, the BIG-IP Controller then marks the server **up**, and resumes sending packets to those services.

The default is 15 seconds.

◆ Note

*If the **timeout_node** interval is shorter than the **timeout_svc** setting, a node can be marked **down** before the services on the node are marked **down**.*

Displaying the current timeout value

Use the following command to display the current timeout setting for node ping:

```
bigpipe timeout_node show
```

Setting a timeout value for node ping

Use the following syntax to set the timeout setting for node ping:

```
bigpipe timeout_node <seconds>
```

The sample command below sets the timeout to 33 seconds.

```
bigpipe timeout_node 33
```

Disabling node ping

To disable node ping, you simply set the node ping timeout value to **0** (zero):

```
bigpipe timeout_node 0
```

WARNING

Node ping is the only form of verification that the BIG-IP Controller uses to determine status of node addresses. If you turn node ping off while one or more node addresses are currently ***down****, the node addresses remain marked* ***down*** *until you turn node ping back on and allow the BIG-IP Controller to verify the node addresses again.*

timeout_svc

```
bigpipe timeout_svc [<port>] show
bigpipe timeout_svc <port> <seconds>
bigpipe timeout_svc <port> 0
```

Description

Sets the amount of time that a specific node has to respond to a service check issued by the BIG-IP Controller. This setting affects three types of service checks:

- Simple service check, where the BIG-IP Controller attempts to establish a connection to the service hosted by the node
- Extended content verification, where the BIG-IP Controller requests specific content from the node
- Extended application verification, where the BIG-IP Controller executes an external service check program that verifies whether or not specific content is available on the node

If a node fails to respond to any type of service check within the specified time, the BIG-IP Controller assumes that the service is down and no longer sends client requests to the service. If the node responds to the next service check, or to subsequent service checks, the BIG-IP Controller marks the service **up**, and resumes sending requests to the service.

WARNING

*The BIG-IP Controller does not attempt to detect the status of a node if node ping is turned off (**bigd -n**) and the **timeout_svc** and **tping_svc** values are set to **0** for a particular node.*

The **timeout_svc** default for each port is set to **0**, which disables service checks on the port.

Note that the BIG-IP Controller monitors only those services that have a **timeout_svc** and **tping_svc** value greater than 0.

Setting the service check timeout

Use the following syntax to set the service check timeout for a specific node port. Note that this setting applies to all nodes that use the port.

```
bigpipe timeout_svc <port> <seconds>
```

For example, the following command sets the service check timeout on port 80 to 120 seconds:

```
bigpipe timeout_svc 80 120
```

Disabling the service check

To disable service check on a specific port, use the above command, but set the <**seconds**> parameter to zero:

```
bigpipe timeout_svc <port> 0
```

Displaying service check timeouts

Use the following command to display the current service check timeout settings for all ports:

```
bigpipe timeout_svc show
```

The system displays the following output:

```
port 80 timeout after 120 seconds
```

The system displays only ports that have a timeout set to a value other than 0.

Use the following syntax to display the current service check timeout setting for a specific port:

```
bigpipe timeout_svc <port> [show]
```

tping_node

bigpipe tping_node show

bigpipe tping_node <seconds>

Description

Sets the interval (in seconds) at which a BIG-IP Controller issues a ping to each server managed by the BIG-IP Controller. If a specific server responds to the ping within a set time, the server is marked **up** and the BIG-IP Controller sends connections to the services hosted by that server. If a server fails to respond to a ping within the specified time, the BIG-IP Controller assumes that the server is no longer available, and it marks the node **down**.

Note that the **timeout_node** setting determines the number of seconds that a server has in which to respond to the ping issued by the BIG-IP Controller.

The default setting for **tping_node** is 5 seconds.

Setting a node ping interval

Use the following syntax to set the number of seconds which a server can take to respond to a ping issued by the BIG-IP Controller:

bigpipe tping_node <seconds>

Disabling node ping

To turn node ping off, set the interval to **0** seconds:

bigpipe tping_node 0

Displaying the current node ping setting

Use the following command to display the current node ping setting:

bigpipe tping_node show

tping_svc

```
bigpipe tping_svc show
bigpipe tping_svc <port> <seconds>
bigpipe tping_svc <port> 0
```

Description

Sets the interval (in seconds) at which BIG-IP Controller issues a service check to one or more specific nodes included in the configuration. This setting affects three types of service check:

- Simple Service check where, the BIG-IP Controller attempts to establish a connection to the service hosted by the node
- Extended Content Verification where, the BIG-IP Controller requests specific content from the node
- Extended Application Verification where, the BIG-IP Controller executes an external service check program that verifies whether or not specific content is available on the node

If a node fails to respond to a service check within the time specified by the **timeout_svc** setting, the BIG-IP Controller marks the service **down**, and no longer routes client requests to it.

WARNING

*The BIG-IP Controller does not attempt to detect the status of a node if node ping is turned off (**bigd -n**) and the **timeout_svc** and **tping_svc** values are set to **0** for a node.*

Setting global service check intervals for a node port

Use the following syntax to set a service check interval for a specific node port:

```
bigpipe tping_svc <port> <seconds>
```

Use the following syntax to turn service check off for a specific node port:

```
bigpipe tping_svc <port> 0
```

Displaying the current service check interval

Use the following syntax to display the intervals at which the BIG-IP Controller issues service checks to all nodes configured for service check:

```
bigpipe tping_svc show
```

treaper

```
bigpipe treaper show
bigpipe treaper <port> <seconds>
bigpipe treaper <port> 0
```

Description

Sets the expiration time for idle TCP connections on a specific port. An idle connection is one in which no data has been received or sent for the number of seconds specified by the **treaper** command. The **treaper** default value is 1005 seconds. For **treaper** to be effective, you should set its value to be greater than the configured timeout for the service daemons installed on your nodes.

The **treaper** command clears the connection tables, avoiding memory problems due to the accumulation of dead, but not terminated, connections.

Setting the idle TCP connection timeout for a virtual port

To set an inactive connection timeout for one or more virtual ports, use the following syntax:

```
treaper <port> <seconds>
```

To turn off the inactive connection timeout, use the same command, but set the number of seconds to zero:

```
treaper <port> 0
```

◆ Note

Typical settings include 120 seconds for 25/SMTP, 120 seconds for 80/www, 300-600 seconds for 20/ftp-data and 21/ftp-data.

Displaying the current inactive connection timeout

To display the current number of seconds that connections are allowed to remain idle before being dropped, use the following syntax:

```
bigpipe treaper show
```

udp

```
bigpipe udp [<port> [...<port>] ] show
bigpipe udp <port> [...<port>] <seconds>
bigpipe udp <port> 0
```

Description

The **udp** command enables UDP traffic on virtual ports and also sets a timeout for idle UDP connections. UDP traffic is enabled only when the timeout is set to a value greater than **0** (zero). You can disable UDP traffic on a port by setting the idle connection timeout to **0** (zero). By default, UDP is disabled on all ports.

Setting the idle connection timeout for UDP traffic

To set the UDP timeout on one or more virtual ports, where the <**seconds**> parameter is the number of seconds before an idle connection is dropped, use the following syntax:

```
bigpipe udp <port> <seconds>
```

For example, the following command sets the UDP timeout to **300** seconds for port 53:

```
bigpipe udp 53 300
```

To turn off UDP timeout for a virtual port, use the above command, setting the <**seconds**> parameter to zero:

```
bigpipe udp <port> 0
```

Displaying UDP settings

Use the following command to display the UDP timeout setting for all ports that allow UDP:

```
bigpipe udp show
```

Use the following syntax to display the timeout setting for a specific virtual port that allows UDP:

```
bigpipe udp <port> show
```

The system displays the output:

```
port 53 idle udp connections expire after 300 seconds <$startrange>
   bigpipe commands: udp;
```

unit

bigpipe unit [show]

bigpipe unit peer [show]

Description

The unit number on a BIG-IP Controller designates which virtual servers use a particular controller in an active-active redundant configuration. You can use the **bigpipe unit** command to display the unit number assigned to a particular BIG-IP Controller. For example, to display the unit number of the unit you are on, type the following command:

```
bigpipe unit show
```

To display the unit number of the other controller in a redundant system, type in the following command:

```
bigpipe unit peer show
```

◆ Note

If you use this command on a redundant system in active/standby mode, the active controller shows as unit 1 and 2, and the standby controller has no unit numbers.

◆ Tip

*The **bigpipe unit peer show** command is the best way to determine whether the respective state mirroring daemons are connected.*

-V

bigpipe -v

Description

Displays the version number of the **bigpipe** command utility.

For example, **bigpipe -v** displays the following output:

```
bigpipe: 3.3
```

version

bigpipe version

Description

Displays the version number of the BIG-IP Controller's operating system.

The **bigpipe version** command provides the following version information:

```
BIG-IP: version 3.3
```

vip

```
bigpipe vip <virt ip>[:<port>] [<ifname>] [unit <ID>] \
   [netmask <ip>] [broadcast <ip>] use pool <pool_name>
bigpipe vip <virt ip>:<port>[/<bitmask>] [<ifname>] [unit <ID>] \
   use pool <pool_name>
bigpipe vip <virt ip>[:<port>] [<ifname>] [unit <ID>] \
   [netmask <ip>] [broadcast <ip>] use rule <rule_name>
bigpipe vip <virt ip>:<port>[/<bitmask>] [<ifname>] [unit <ID>] \
   use rule <rule_name>
bigpipe vip [<virt ip>[:<port>]] [...<virt ip>[:<port>] ] show
bigpipe vip <virt ip>[:<port>] [<ifname>] [ ... <virt ip>[:<port>]\
   ] enable | disable | delete
bigpipe vip <virt ip>[:<port>] [... <virt ip>[:<port>]] limit \
   <max conn>
bigpipe vip <virt ip>:<port> translate port enable | disable | show
bigpipe vip <virt ip>:<port> translate addr enable | disable | show
bigpipe vip <virt ip>:<port> lasthop pool <pool_name> | none | show
bigpipe vip <virt ip>:<port> mirror conn enable | disable | show
bigpipe vip [<virt ip:port>] stats reset
bigpipe vip <ip>:<port> accelerate disable
bigpipe vip <ip>:<port> use pool the_pool accelerate disable
```

Description

Creates, deletes, and displays information about virtual servers. This command also sets connection mirroring, connection limits, and timeouts on a virtual server.

Defining a virtual server

Virtual servers are port-specific, and if you are configuring a site that supports more than one service, you need to configure one virtual server for each service offered by the site. Use the following syntax to define the pools or rules to which a virtual server maps. The **unit** <**ID**> parameter specifies which unit handles the virtual

server in an active-active redundant configuration. You can associate pools or rules with a virtual server. The following sections describe the syntax for associating a pool or a rule with a virtual server.

Configuring a virtual server to use a load balancing pool

Use the following syntax to create a virtual server that references a load balancing pool. Note that you must create a pool before you can create a virtual server that references the pool. For information about creating a pool, see *Creating a pool*, on page 2-36.

```
bigpipe vip <virt ip>:<port> [ifname] [unit <ID>] use pool
   <pool_name>
```

For example, if you want to create a virtual server that references the pool **my_pool**, the command might look like this:

```
bigpipe vip 11.12.1.53:80 use pool my_pool
```

Configuring a virtual server to use a load balancing rule

Use the following syntax to create a virtual server that references a load balancing rule. Note that you must create a rule before you can create the virtual server that references the rule. For information about creating a rule, see *Associating a rule with a virtual server*, on page 2-53.

```
bigpipe vip <virt ip>:<port> [ifname] [unit <ID>] use rule
   <rule_name>
```

For example, if you want to create a virtual server that references the rule **my_rule**, the command might look like this:

```
bigpipe vip 11.12.1.53:80 use pool my_rule
```

Displaying information about virtual servers

Use the following syntax to display information about all virtual servers included in the configuration:

bigpipe vip show

Use the following syntax to display information about one or more virtual servers included in the configuration:

```
bigpipe vip <virt ip>:<port> [...<virt ip>:<port>] show
```

The command displays information such as the nodes associated with each virtual server, the nodes' status, and the current, total, and maximum number of connections managed by the virtual server since the BIG-IP Controller was last rebooted.

Defining an interface for a virtual server

If you have multiple external (destination processing) interfaces, you can specify one of them when you define a virtual server.

- If you specify an interface name, the BIG-IP Controller responds to ARP requests for the virtual address on that interface.
- If you do not specify an interface name, the BIG-IP Controller responds to ARP requests for the virtual server on the default interface.
- If you do not want the BIG-IP Controller to respond to ARP requests on any interface, use the option **none** in place of the an <**ifname**> parameter.

All virtual servers that share a virtual address must use the same external interface. Changing the interface for a virtual server changes the interface for all virtual servers that have the same virtual address.

Setting a user-defined netmask and broadcast

The default netmask for a virtual address, and for each virtual server hosted by that virtual address, is determined by the network class of the IP address entered for the virtual server. The default broadcast is automatically determined by the BIG-IP Controller, and it is based on the virtual address and the current netmask. You can override the default netmask and broadcast for any virtual address.

All virtual servers hosted by the virtual address use the netmask and broadcast of the virtual address, whether they are default values or they are user-defined values.

If you want to use a custom netmask and broadcast, you define both when you define the virtual server:

```
bigpipe vip <virt ip>[:<port>] [<ifname>] [netmask <ip>] \
   [broadcast <ip>] use pool <pool_name>
```

◆ Note

The BIG-IP Controller calculates the broadcast based on the IP address and the netmask. A user-defined broadcast address is not necessary.

Again, even when you define a custom netmask and broadcast in a specific virtual server definition, the settings apply to all virtual servers that use the same virtual address. The following sample command shows a user-defined netmask and broadcast:

```
bigpipe vip www.SiteOne.com:http netmask 255.255.0.0 \
   broadcast 10.0.140.255 use pool my_pool
```

The **/bitmask** option shown in the following example applies network and broadcast address masks. In this example, a 24-bit bitmask sets the network mask and broadcast address for the virtual server:

```
bigpipe vip 206.168.225.1:80/24 use pool my_pool
```

You can generate the same broadcast address by applying the 255.255.255.0 netmask. The effect of the bitmask is the same as applying the 255.255.255.0 netmask. The broadcast address is derived as 206.168.225.255 from the network mask for this virtual server.

Setting a connection limit

The default setting is to have no limit to the number of concurrent connections allowed on a virtual server. You can set a concurrent connection limit on one or more virtual servers using the following command:

```
bigpipe vip <virt ip>[:<port>] [...<virt ip>[:<port>] ] limit \
   <max conn>
```

The following example shows two virtual servers set to have a concurrent connection limit of 5000 each:

```
bigpipe vip www.SiteOne.com:http www.SiteTwo.com:ssl limit 5000
```

To turn off the limit, set the <**max conn**> variable to zero:

```
bigpipe vip <virt ip>[:<port>] [...<virt ip>[:<port>] ] limit 0
```

Setting translation properties for virtual addresses and ports

Turning port translation off for a virtual server is useful if you want to use the virtual server to load balance connections to any service. Use the following syntax to enable or disable port translation for a virtual server.

```
bigpipe vip <virt ip>:<port> translate port enable | disable | show
```

You can also configure the translation properties for a virtual server address. This option is useful when the BIG-IP Controller is load balancing devices that have the same IP address. This is typical with the nPath routing configuration where duplicate IP addresses are configured on the loopback device of several servers. Use the following syntax to enable or disable address translation for a virtual server.

```
bigpipe vip <virt ip>:<port> translate addr enable | disable | show
```

Setting up last hop pools for virtual servers

In cases where you have more than one router sending connections to a BIG-IP redundant system, you may want to route connections back through the same router from which they were received. To configure a last hop pool, you must first create a pool that contains the routers for the BIG-IP redundant system. After you create a router pool, use the following syntax to configure a last hop pool for a virtual server.

```
bigpipe vip <virt ip>:<port> lasthop pool <pool_name> | none | show
```

Mirroring connection information

Mirroring provides seamless recovery for current connections and for when a BIG-IP Controller fails. When you use the mirroring feature, the peer controller maintains the same current connection and persistence information as its partner controller. Transactions such as FTP file transfers continue as though uninterrupted.

To control mirroring for a virtual server, use the **mirror** command to enable or disable mirroring of connections. The syntax of the command is:

```
bigpipe vip <virt ip>:<port> mirror conn enable | disable
```

To print the current mirroring setting for a virtual server:

```
bigpipe vip <virt ip>:<port> mirror conn show
```

If you do not specify **conn**, the BIG-IP Controller displays all mirrored connection information.

◆ Note

If you set up mirroring on a virtual server that supports FTP connections, you need to mirror the control port virtual server, and the data port virtual server.

The following example shows the two commands used to enable mirroring for virtual server **v1** on the FTP control and data ports:

```
bigpipe vip v1:21 mirror conn enable
bigpipe vip v1:20 mirror conn enable
```

Removing and returning a virtual server to service

You can remove an existing virtual server from network service, or return the virtual server to service, using the **disable** and **enable** keywords. When you disable a virtual server, the virtual server no longer accepts new connection requests, but it allows current connections to finish processing before the virtual server goes **down**. Use the following syntax to remove a virtual server from network service:

```
bigpipe vip <virt ip>:<port> [...<virt ip>:<port>] disable
```

Use the following syntax to return a virtual server to network service:

```
bigpipe vip <virt ip>:<port> enable
```

Removing and returning a virtual address to service

You can remove an existing virtual address from network service, or return the virtual address to service, using the **disable** and **enable** keywords. Note that when you enable or disable a virtual address, you inherently enable or disable all of the virtual servers that use the virtual address.

```
bigpipe vip <virt ip> disable
```

Use the following syntax to return a virtual address to network service:

```
bigpipe vip <virt ip> enable
```

Displaying information about virtual addresses

You can also display information about the virtual addresses that host individual virtual servers. Use the following syntax to display information about one or more virtual addresses included in the configuration:

```
bigpipe vip <virt ip> [... <virt ip> ] show
```

The command displays information such as the virtual servers associated with each virtual address, the status, and the current, total, and maximum number of connections managed by the virtual address since the BIG-IP Controller was last rebooted, or since the BIG-IP Controller became the active unit (redundant configurations only).

Deleting a virtual server

Use the following syntax to permanently delete one or more virtual servers from the BIG-IP Controller configuration:

```
bigpipe vip <virt ip>:<port> [... <virt ip>:<port>] delete
```

Resetting statistics for a virtual server

Use the following command to reset the statistics for an individual virtual server:

```
bigpipe vip [<vip ip:port>] stats reset
```

Turning software acceleration off for virtual servers using IPFW rate filters

Additional enhancements are included in this release that speed packet flow for TCP connections when the packets are not fragmented. In most configurations these software enhancements are automatically turned on and do not require any additional configuration.

However, you may want to turn off these enhancements for individual virtual servers that use IPFW rate filters. With the speed enhancements on, IPFW only examines the first SYN packet in any given connection. If you want to filter all packets, you should turn off the speed enhancements. To do this, you must first set the global state of the system on, and then you must turn off the feature for individual virtual servers that use IPFW rate filtering. You can change the settings for these enhancements from the command line or in the Configuration utility.

Setting software acceleration controls from the command line

Before you can turn off software acceleration for a virtual server, you must set the **sysctl** variable **bigip.fastpath_active** to on (2) with the following command:

```
sysctl -w bigip.fastpath_active=2
```

After you set the **sysctl** variable, use the following **bigpipe** commands to disable software acceleration for existing virtual servers that use IPFW rate filtering:

```
bigpipe vip <ip>:<port> accelerate disable
```

For example, if you want to turn acceleration off for the virtual server 10.10.10.50:80, type the following command:

```
bigpipe vip 10.10.10.50:80 accelerate disable
```

You can define a virtual server with acceleration disabled using the following syntax:

```
bigpipe vip <ip>:<port> use pool the_pool accelerate disable
```

For example, if you want to define the virtual server 10.10.10.50:80 with the pool **IPFW_pool** and acceleration turned off, type the following command:

```
bigpipe vip 10.10.10.50:80 use pool IPFW_pool accelerate disable
```

Backward-compatible commands

The following **bigpipe** commands from previous versions of the BIG-IP Controller are compatible with this version and have been included here for users of previous versions of the product.

```
dt [<ip>[:<port> ] ]

port <port> [<port>... ] [allow | deny] [ limit <limit> ]

vip <virt ip>:<port> persistmask [ <IP address mask> ]

vip <virt ip>:<port> persistmask [ none | show ]

vip <virt ip>[:<port>] [<ifname>] netmask <ip> \
   [ broadcast <ip> ] define <node ip>[:<port> \
   [ <node ip>[:<port>... ] [ special ssl <value> <value> ]

nat <node ip> to <NAT ip> [<ifname>] netmask <ip> \
   [ broadcast <ip> ]

fo [ master | slave ]

vip <virt ip>[:<port>] [/<bitmask>] [<ifname>|none ] \
   [unit <unit ID>] define <node ip>[:<port>] \
   [..<node ip>[:<port>] ] [special ssl <seconds> <seconds>]

vip <virt ip>[:<port>] netmask <ip> [broadcast <ip>] \
   [<ifname> | none ] [unit <unit ID>] define <node ip>[:<port>] \
   [...<node ip>[:<port>] ] [special ssl <seconds> <seconds>] \
   [special cookie insert | rewrite | passive <days>d <hh:mm:ss>]]

vip <virt ip>[:<port>] netmask <ip> [broadcast <ip>] \
   [<ifname> | none ] [unit <unit ID>] define <node ip>[:<port>] \
   [...<node ip>[:<port>] ] [special cookie hash <name> <offset>
   <length>]
```

```
vip <virt ip>:<port> mirror persist enable | disable | show

vip <virt ip>:<port> persist show | dump | <value>

vip <virt ip>:<port> persist mask <ip> | none | show

vip 0.0.0.0:<port> sticky [ enable | disable | show | clear | dump ]

vip 0.0.0.0:<port> sticky mask [ <ip> | none | show ]

vip sticky dump

vip sticky clear
```

F5 Networks BIG-IP Local Traffic Controller

3

Configuration Files

Configuration files for the BIG-IP Controller

File	Description
/etc/bigip.conf	Stores virtual server and node definitions and settings, including node ping settings, the load balancing mode, and NAT and SNAT settings.
/etc/bigconf.conf	Stores the user preferences for the Configuration utility.
/etc/bigd.conf	Stores service check settings.
/etc/hosts.allow	Stores the IP addresses of workstations that are allowed to make administrative shell connections to the BIG-IP Controller.
/etc/netstart	Stores basic system start up settings.
/etc/ipfw.conf	Stores IP filter settings.
/etc/rateclass.conf	Stores rate class definitions.
/etc/ipfwrate.conf	Stores IP filter settings for filters that also use rate classes.
/etc/snmpd.conf	Stores SNMP configuration settings.
/etc/bigip.license	Stores authorization information for the BIG-IP Controller.
/etc/syslog.conf	Stores the configuration files for **syslogd** under the BIG-IP Controller.
/etc/rc.sysctl	Stores the default UNIX and the BIG-IP Controller ***sysctl*** variables.
/etc/hosts	Stores the hosts table for the BIG-IP Controller.
/etc/rc.local	Stores the local daemons, filters, and local boot settings for the BIG-IP Controller.
/etc/irs.conf	Controls information retrieval functions in the C library.
/etc/login.conf	UNIX system file, modified for the BIG-IP Controller.
/etc/rc	UNIX system startup script, modified for the BIG-IP Controller.
/etc/sshd_config	This is the configuration file for the secure shell server (SSH). It contains all the access information for people trying to get into the system via SSH.
/etc/wideip.conf	This is a 3-DNS Controller configuration file. For more information, please refer to the documentation for that product.
/VENDOR	This file contains information describing F5 Networks. It includes the company name, a common name, contact information, and the text of the licensing agreement for the software.

File	Description
/VERSION	Contains name of the product, the number, and the access rights (BIG-IP 3.3 HA, for example).
/usr/contrib/bin/ssh-askpass	This is the external program used by the SSH configuration utility to ask the user for his password from an X-windows system. It allows **SSH** to connect to a remote site, or generate a PPKey pair, in a secure manner.
/var/f5/httpd/conf/cert.conf	The information for the public key/private key certification infrastructure for the webserver.
/var/f5/httpd/conf/httpd.conf	The main configuration file for the webserver.
/var/f5/httpd/ssl/lib/ssleay.cnf	This file holds the configuration information for how the SSL library interacts with browsers, and how key information is generated.
/var/f5/httpd/ssl/lib/ssleay.conf	This file holds the configuration information for how the SSL library interacts with browsers, and how key information is generated.
/var/f5/httpd/basicauth/users	The webserver password file. Contains the user names and passwords of the people permitted to access whatever is provided by the webserver.
/var/f5/bigdb/user.db	This is the location of the BIG/db database. This database holds various configuration information.

F5 Networks BIG-IP Local Traffic Controller

4

BIG-IP Controller Configuration Utilities

Introducing the BIG-IP Controller configuration utilities

The BIG-IP Controller includes a number of configuration utilities. These utilities allow you to reconfigure portions of your installation after initial configuration.

The following configuration utilities are available on the BIG-IP Controller:

- **config**
  This utility is also known as the First-Time Boot utility. This utility runs all the other utilities required to configure the BIG-IP Controller, most of the utilities in this list.
- **config_failover**
  Use this utility to prepare a new redundant system for the commands that synchronize redundant controllers.
- **reconfig-httpd**
  Use this utility to reconfigure the web server on the BIG-IP Controller.
- **config_telnetd**
  Use this utility to configure the Telnet and FTP daemons.
- **config_sshd**
  Use this utility to configure the SSH daemon.
- **config_rshd**
  Use this utility to configure the RSH daemon.

config

Description

This utility starts automatically the first time you boot up a BIG-IP Controller. The **config** utility, referred to as the First-Time Boot utility, is a wizard that walks you through a brief series of required configuration tasks. These tasks include defining a root password and configuring IP addresses for the interfaces. You can run the First-Time Boot utility to reconfigure a controller.

The First-Time Boot utility is organized into three phases: configure, confirm, and commit. Each phase walks you through a series of screens, presenting the information in the following order:

- Root password
- Host name
- Default route (typically a router's IP address)
- Time zone
- DNS proxying
- Interface settings for the each network interface
- Configuration for BIG-IP Controller redundant systems (fail-over IP address)
- IP address for remote administration
- Remote administration access for vendor support
- Settings for the BIG-IP web server
- Web server administration access for vendor support

When using the config utility, you first configure all of the required information, then you have the opportunity to confirm each individual setting or correct it if necessary, and finally your confirmed settings are committed and saved to the system. Note that the screens you see are tailored to the specific hardware and software configuration that you have.

If you have a stand-alone system, for example, the First-Time Boot utility skips the redundant system screens.

To run the First-Time Boot utility, type in the following command:

```
config
```

config_failover

Description

You must run the **config_failover** script on the second controller in a redundant system in order to share keys with the peer BIG-IP Controller.

The script prompts you for the root password of the other controller in the redundant system. After confirming your input, the **config_failover** script attempts to access the peer system and configure both systems to communicate with one another. This provides the secure communication channel the controllers use to exchange configuration data when you run the **bigpipe configsync** option, or use the **Config Sync** button in the Configuration utility.

To run the **config_failover** script, type the following command on the command line:

```
config_failover
```

reconfig-httpd

Description

Use the **reconfig-httpd** configuration utility to reconfigure the HTTPD server on a BIG-IP Controller.

This script prompts you for an IP address from which administrators can access the BIG-IP Controller through the BIG-IP web server. You can use wildcard characters (*****) to include all addresses from a specific part of the network. This utility also prompts you to create a support account for access by technical support.

If the service port for the BIG-IP web server (**httpd**) is closed, this script automatically opens the service port to permit access to the BIG-IP web server.

To run the BIG-IP web server configuration utility, type the following command:

```
reconfig-httpd
```

config_telnetd

Description

Use the **config_telnetd** configuration script to configure the Telnet and FTP servers on a BIG-IP Controller. The script prompts you to configure each service independently. This allows you to enable Telnet but not FTP, for example.

The script prompts you for a configuration address for each service from which administrators may access the BIG-IP Controller. You can use wildcard characters (*****) to include all addresses from a specific part of the network. This utility also prompts you to create a support account for access by technical support.

If **inetd** is not currently configured, this script configures **inetd** for the requested services. If the ports for Telnet or FTP are closed, this script opens the ports to permit Telnet or FTP connections to the BIG-IP Controller.

To run the Telnet/FTP configuration utility, type in the following command:

```
config_telnetd
```

◆ Note

*Running **config_telnetd** again replaces the current configuration.*

config_sshd

Description

Use this utility to configure the secure shell server (**sshd**) on a BIG-IP Controller. This utility prompts you for an IP address from which administrators may access the BIG-IP Controller with SSH. You can use wildcard characters (*****) to include all addresses from a specific part of the network. This utility also prompts you to create a support account for access by technical support.

If the service port for SSH is closed, this script opens the service port to permit SSH connections to the BIG-IP Controller.

To run the secure shell configuration utility, type in the following command:

```
config_sshd
```

◆ Note

*Re-running **config_sshd** again replaces the current configuration.*

config_rshd

Description

Use the **config_rshd** configuration utility to configure the remote shell (**rshd**) server on a BIG-IP Controller. This utility prompts you for an IP address from which administrators may access the BIG-IP Controller. You can use wildcard characters (*****) to include all addresses from a specific part of the network. This utility also prompts you to create a support account for access by technical support.

If **inetd** is not currently configured, this script configures **inetd** for the remote shell server (**rshd**). If the service port for **rsh** is closed, this utility opens the service port to permit **rsh** connections to the BIG-IP Controller.

To run the **rsh** configuration utility, type in the following command:

```
config_rshd
```

Note

*Running **config_rshd** again replaces the current configuration.*

F5 Networks

BIG-IP®

Local Traffic Controller

5

BIG-IP System Control Variables

Setting BIG-IP system control variables

The BIG-IP Controller hardware and software boot up with a configuration specified, in part, by the system control variables stored in the **/etc/rc.sysctl** file. Most of these variables are standard BSD UNIX system control variables, while some are used exclusively by the BIG-IP Controller. In most cases, a variable is just toggled **off (0)** or **on (1)**, but some variables may also store specific values, such as a port number.

You can use three methods to set system control variables affecting the BIG-IP Controller:

- **The Configuration utility**
  Navigate to a system control variable and edit it in the browser with the Configuration utility.
- **sysctl command**
  Write system control variable values directly to **/etc/rc.sysctl** using this command line utility.
- **vi**
- **pico**
  Use a text editor, such as **vi** or **pico**, to edit **/etc/rc.sysctl** directly.

Following are all the system control variables you will need.

sysctl

```
sysctl -a
sysctl <variable name>
sysctl -w <variable name>=<value>
```

Displaying current system control variable settings

To display the settings of all system control variables, use the following syntax:

```
sysctl -a
```

To display the current setting for an individual variable, use the following command syntax:

```
sysctl <variable name>
```

Setting a system control variable

Use the following syntax to write a value for a system control variable in **/etc/rc.sysctl**:

```
sysctl -w <variable name>=<value>
```

For example, the following command sets vipnoarp mode to **on** at boot:

```
sysctl -w bigip.vipnoarp=1
```

To turn vipnoarp mode **off** at boot, you would write the setting to **/etc/rc.sysctl** using the following command:

```
sysctl -w bigip.vipnoarp=0
```

bigip.bonfire_mode

Description

bigip.bonfire_mode=1

Sets the BIG-IP Controller to operate in Transparent Node mode, where it can perform load balancing on routers and router-like devices, such as transparent firewalls.

bigip.bonfire_mode=0

(Default) Transparent Node mode is **off**.

Note

With this version of the BIG-IP Controller, Transparent Node mode is no longer necessary. You do not need to set this variable. This variable only exists for backward compatibility. You can define a virtual server with address translation turned ***on*** *or* ***off*** *at any time. For more information about address translation, see the* ***BIG-IP Administrator Guide****.*

bigip.bonfire_compatibility_mode

Description

bigip.bonfire_compatibility_mode=1

Turns off port translation on the BIG-IP Controller. This is useful if a node port is only being used to specify a service check port.

bigip.bonfire_compatibility_mode=0

(Default) Port translation is **on**.

Note

With this version of the BIG-IP Controller, Transparent Node mode is no longer necessary. You do not need to set this variable. This variable only exists for backward compatibility. You can define a virtual server with port translation turned ***on*** *or* ***off*** *at any time. For more information about port translation, see the* ***BIG-IP Administrator Guide****.*

bigip.fastest_max_idle_time

Description

bigip.fastest_max_idle_time=<seconds>

Sets the number of seconds a node can be left idle by the **fastest** load balancing mode. This forces the BIG-IP Controller to send fewer connections to a node that is responding slowly. This allows the BIG-IP Controller to periodically recalculate the response time of the slow node.

bigip.fastpath_active

You can use this variable to control additional enhancements that speed packet flow for TCP connections when the packets are not fragmented. In most configurations these software enhancements are automatically turned on and do not require any additional configuration.

However, you may want to turn off these enhancements for individual virtual servers that use IPFW rate filters. With the speed enhancements **on**, IPFW only examines the first SYN packet in any given connection. If want to filter all packets, you should turn the speed enhancements **off**. To do this, you must first set the global state of the system **on**, and then you must turn the feature **off** for individual virtual servers that use IPFW rate filtering. You can change the settings for these enhancements from the command line or in the Configuration utility.

Setting software acceleration controls from the command line

There are three global states you can set with the **sysctl** variable **bigip.fastpath_active**. The default state is automatic, or a value of **1**. The global states are:

```
0 = off
1 = automatic
2 = on
```

The additional speed enhancements are globally disabled if the sysctl variable **bigip.fastpath_active** is **off (0)** or if **bigip.fastpath_active** is set to **automatic (1)** and an IPFW rate filter exists in the configuration.

To provide the benefits of software acceleration for virtual servers that do not use rate filtering and turn off software acceleration for virtual servers that use IPFW rate filtering, you can set the **sysctl** variable **bigip.fastpath_active** to **on (2)** with the following command:

```
sysctl -w bigip.fastpath_active=2
```

After you set the **sysctl** variable, use the following **bigpipe** command to disable software acceleration for virtual servers that use IPFW rate filtering:

```
bigpipe vip <ip>:<port> accelerate disable
```

bigip.forwarding_vip_overrides_default_snat

bigip.forwarding_vip_overrides_default_snat=1

Setting this variable to **1** turns off SNAT for forwarding virtual servers. The result is that the default SNAT ignores a new connection with a destination that matches a forwarding virtual server. This causes outbound connections to use either a forwarding virtual server or the default SNAT depending on the destination of the packet that initiates the connection. The default setting for this variable is **1**.

bigip.forwarding_vip_overrides_default_snat=0

Setting this variable to **0** leaves SNAT on for forwarding virtual servers. The result is that both a SNAT and a virtual server connection need to be created by an outbound packet initiating a connection to the destination specified by a forwarding virtual server.

bigip.halt_reboot_timeout

Description

bigip.halt_reboot_timeout=2

This value is the number of seconds the BIG-IP Controller can stop during boot up before the watchdog card hard reboots the controller. The default value for this setting is **2** seconds.

bigip.improved_fastest

bigip.improved_fastest=1

To use the improved Fastest, Observed, and Predictive load balancing modes, this variable must be set to **1**. The default value for this variable is **1**.

bigip.improved_fastest=0

If you do not want to use the improved fastest modes, set this variable to **0**. You might want to do this in a case where all connections to a node address are sequential and the application response is variable, slow, and unrelated to transport protocol response.

bigip.max_sticky_entries

Description

bigip.max_sticky_entries=2048

This is the maximum number of sticky entries allowed to accumulate on the BIG-IP Controller when using destination address affinity (sticky persistence). When the maximum value is reached, the BIG-IP Controller stops accumulating sticky entries. The default value for this entry is **2048**.

bigip.memory_reboot_percent

Description

bigip.memory_reboot_percent=0

The default value for this variable is **0**, which is disabled. The minimum value for this variable is **80**, or 80 percent of the total physical memory on the controller. The value you type, 80 or higher, is the percentage of memory that is in use before the BIG-IP Controller automatically reboots.

bigip.open_3dns_lockdown_ports

Description

bigip.open_3dns_lockdown_ports=0

(Default) This variable is only required when running a 3-DNS Controller. This variable is set to **0** on the BIG-IP Controller when the 3-DNS Controller is not present in the network configuration. (See the ***3-DNS Administrator Guide*** for more information.)

bigip.open_telnet_port

Description

bigip.open_telnet_port=1

Opens the Telnet port (23) to allow administrative Telnet connections. (Useful for an international BIG-IP Controller, or for a controller that needs to communicate with non-crypto 3-DNS Controllers.)

bigip.open_telnet_port=0

Opens the Telnet port to allow administrative Telnet connections (useful for non-crypto BIG-IP Controllers).

bigip.open_ftp_ports

Description

bigip.open_ftp_ports=1

Opens the FTP ports (20 and 21) to allow administrative FTP connections (useful for international BIG-IP Controllers).

bigip.open_ftp_ports=0

The default setting for this variable is **0**. The FTP port does not allow administrative FTP connections.

bigip.open_ssh_port

Description

bigip.open_ssh_port=1

Opens the SSH port (22) to allow administrative connections (useful only for US BIG-IP Controllers).

bigip.open_ssh_port=0

The default setting for this variable is **0**. The SSH port does not allow administrative connections.

bigip.open_rsh_ports

Description

bigip.open_rsh_ports=1

Opens the RSH ports (512, 513, and 514) to allow RSH connections (useful for international BIG-IP Controllers, or on US controllers that need to communicate with international 3-DNS Controllers).

bigip.open_rsh_ports=0

The default setting for this variable is **0**. The RSH port does not allow RSH connections.

bigip.persist_map_proxies

bigip.persist_map_proxies=1

The default setting for the map proxies for the persistence variable is **on**. The AOL proxy addresses are hard-coded in this release. This enables you to use client IP address persistence with a simple persist mask, but forces all AOL clients to persist to the same server. All AOL clients will persist to the node that was picked for the first AOL client connection received.

The class B networks, 195.93 and 205.188, are mapped to 152.163 for persistence. For example, client 195.93.3.4 would map to 152.63.3.4 for persistence records only. This mapping is done prior to applying the persist mask. Use **bigpipe vip persist dump** to verify that the mapping is working.

We recommend that in addition to setting this **sysctl** variable, you set a persist mask of 255.255.0.0 so that all the AOL addresses map to a common address. For example, Table 5.1 is an example of how setting this variable and a persist mask of 255.255.0.0 would map a sample set of client addresses.

Sample Client Address	Persist Address
152.44.12.3	195.93.0.0
152.2.99.7	195.93.0.0
170.11.19.22	195.93.0.0
202.67.34.11	195.93.0.0
205.188.11.2	195.93.0.0
208.33.23.4	208.33.0.0 (non AOL address is not mapped)

Table 5.1 *Address mapping of sample clients*

bigip.persist_map_proxies=0

Set this variable to **0** to turn this variable **off**.

bigip.persist_time_used_as_limit

Description

bigip.persist_time_used_as_limit=0

(Default) Forces the persistent connection timer to reset on each packet for persistent sessions.

bigip.persist_time_used_as_limit=1

Resets timer only when the persistent connection is initiated.

Note

For SSL persistence, the timer is always reset on each packet.

bigip.persist_on_any_vip

Description

bigip.persist_on_any_vip=1

All simple persistent connections from the same client IP address are sent to the same node (matches the client IP address but not the virtual address or virtual port the client is using).

bigip.persist_on_any_vip=0

The default setting for this variable is **off**.

bigip.persist_on_any_port_same_vip

Description

bigip.persist_on_any_port_same_vip=1

All simple persistence connections from a client IP address that go to the same virtual address also go to the same node (matches the client address and the virtual IP address but not the virtual port).

bigip.persist_on_any_port_same_vip=0 The default setting for this variable is **off**.

bigip.tcphps_mss_override

Description

bigip.tcphps_mss_override=(<1460)

Allows you to decrease the default maximum segment size (MSS) from 1460 to a smaller value. The defined value is the value announced to clients by the TCP server proxy on the BIG-IP Controller in the SYN/ACK packet.

bigip.tcphps_mss_override=0

(Default) The BIG-IP Controller requests the MSS from the node when negotiating connections on the node's behalf.

bigip.verbose_log_level

Description

bigip.verbose_log_level=0

Turns port denial logging **off**. No messages are logged.

bigip.verbose_log_level=1

Turns UDP port denial logging **on**. This logs UDP port denials to the BIG-IP Controller address.

bigip.verbose_log_level=2

Turns TCP port denial logging **on**. This logs TCP port denials to the BIG-IP Controller address.

bigip.verbose_log_level=4

Turns virtual UDP port denial logging **on**. This logs UDP port denials to the virtual server address.

bigip.verbose_log_level=8

Turns virtual TCP port denial logging **on**. This logs TCP port denials to the virtual server address.

bigip.verbose_log_level=15

Turns TCP and UDP port denial logging **on**. This logs TCP and UDP port denials to the virtual server address and the BIG-IP Controller address. Setting this variable to **15** turns on logging levels **1**, **2**, **4**, and **8**.

bigip.vipnoarp

Description

bigip.vipnoarp=1

Prevents the BIG-IP Controller from issuing ARP requests when rebooted. This is useful for configurations that contain 1,000 or more virtual servers. This setting also prevents you from configuring virtual servers as IP addresses on the BIG-IP Controller external interface.

bigip.vipnoarp=0

The default setting for this variable is **0**. The BIG-IP Controller issues ARP requests on reboot.

bigip.webadmin_port

Description

bigip.webadmin_port=443

Specifies the port number used for administrative web access. The default port for web administration is port **443**.

net.inet.ip.forwarding

Description

net.inet.ip.forwarding=1

Exposes node IP addresses on the internal network, allowing clients to connect directly to nodes, and also allows nodes to initiate connections with computers external to the BIG-IP Controller. Typically, this setting is used only on systems that cannot use NATs (for example, a network that uses CORBA or the NT Domain).

net.inet.ip.forwarding=0

(Default) IP forwarding is **off**.

net.inet.ip.sourcecheck

Description

net.inet.ip.sourcecheck=1

This setting enables the BIG-IP Controller to check the source IP address of incoming packets before it checks the packet for other information (for example, the virtual server).

Source checking tries to allocate a route back to the source of the packet, and if the route cannot be found, or if the route of the interface is on an interface different from the interface from which the packet was received, the packet is discarded. Each time a packet is discarded, the **bad source interface** counter is incremented.

net.inet.ip.sourcecheck=0

The default setting for this variable is IP source checking is **0** (off).

F5 Networks BIG-IP Local Traffic Controller

6

System Utilities

Introducing the BIG-IP Controller system utilities

The BIG-IP Controller has three main system utilities. These utilities control fail-over functions, monitor pool members, and communicate with the F5 Networks 3-DNS Controller. This chapter describes the following utilities:

sod
The switch over daemon (**sod**) controls the fail-over functions of the BIG-IP Controller.

bigd
The **bigd** monitors services and nodes for the BIG-IP Controller. This daemon determines whether nodes and services are healthy.

big3d
The **big3d** answers queries from the F5 Networks 3-DNS Controller.

sod

sod [-help] [-tty00] [-tty01]

SOD Option	Description
-help	Prints help text.
-tty00	Use **tty0** for fail-over monitoring.
-tty01	Use **tty1** for fail-over monitoring.

Description

The switch-over daemon (**sod**) controls the BIG-IP Controller fail-over functions. It has a command line interface for some functions.

Command line usage

The **sod** daemon is used as a command line utility for some of its functions.

To display the online help for **sod** use this command:

```
sod -help
```

Daemon start up options

The **sod** daemon is configured in **/etc/rc.local**. You can configure the **sod** daemon in two ways:

- Serial port(s) used for hardware fail-over cable connections
- Forced fail-over role (active or standby) at boot

Note

*Every time you change your **sod** daemon configuration, you need to reboot the BIG-IP Controller.*

Fail-over cable port configuration in sod startup

The **sod** daemon startup line in **/etc/rc.local** accepts two optional parameters: **-tty00** and **-tty01**. These parameters specify which of the two 9-pin serial ports (one of them may be a 25-pin serial port on older BIG-IP Controller models) is used as the fail-over cable connection. The default is **-tty01**. Use one (or none) of the **-tty***xx* options to configure a single fail-over cable. Use both options to configure two cables (redundant fail-over cables), as in the following example:

```
echo " sod (and bigd)."; /sbin/sod -tty00 -tty01 -- \
   bigd ${bigdflags} 2> /dev/null
```

References to these fail-over cable connection ports in the **sod** startup line in **/etc/rc.local** are always made using the UNIX device name, while the hardware and BIOS settings for the ports use COM and serial port designations, respectively.

BIOS	COM	UNIX
Port 2 (2f8 irq 3)	COM2	/dev/tty01
Port 1 (3f8 irq 4)	COM1	/dev/tty00

Table 6.1 *9-pin serial port designations in BIOS, hardware, and UNIX operating systems.*

◆ Note

The 9-pin serial port labeled "Terminal" on the back of the BIG-IP Controller is COM2.

bigd

bigd [-d filename] [-n] [-s] [-v] [-V]

Description

This daemon monitors services and nodes for the BIG-IP Controller. The **bigd** daemon provides service check functions for simple (node ping), extended content verification, and extended application verification service checks. Usage is supported for cases where the check port for a node is not the same as the node port. Table 6.2 contains the options available for **bigd**.

Option	Description
-d filename	Check syntax of the specified configuration file, and then exit. This option cannot be used in conjunction with any other option.
-n	Do not ping nodes.
-s	TCP node ping (default is ICMP)
-v	Print version number.
-V	Print verbose output to message logs.

Table 6.2 *The* ***bigd*** *options*

Files

/etc/bigip.conf

/etc/rc.local

/etc/bigd.conf

/var/log/bigd

/var/log/messages

Configuring bigd

The configuration files in Table 6.3 contain configuration information for **bigd**.

File	Description
/etc/rc.local	Starts **bigd** with the options specified.
/etc/bigip.conf	Contains configuration information for **timeout_svc** and **tping_svc**.
/etc/bigd.conf	Contains configuration information for ECV and EAV service checks.

***Table 6.3 bigd** configuration files*

Starting bigd

The standard way to start **bigd** is by configuring the **sod** startup line in **/etc/rc.local**:

```
echo " sod (and bigd)."; /sbin/sod -- bigd ${bigdflags} /
   2> /dev/null
```

This syntax starts **bigd** after the boot configuration in **/etc/bigip.conf** has been loaded and started. This is the optimal sequence for startup if you use ping aliases. If **bigd** is started before **sod** when ping aliases are defined, node pinging starts before ping aliases have been loaded.

You can also start and restart **bigd** on the command line with options:

```
bigd
```

This is the best way to restart **bigd** if you make changes to the **/etc/bigd.conf** file. This method stops any existing **bigd** processes and restarts the daemon using the configuration in **/etc/rc.local** and **/etc/bigd.conf**.

Setting the node ping parameters used by bigd

Node ping uses the **timeout_node** and **tping_node** parameters (set in **/etc/bigip.conf**) to set the length of time between pings and the length of time to wait for a ping response before timeout.

Setting service check parameters used by bigd

Simple and extended service checks use the **timeout_svc** and **tping_svc** parameters (set in **/etc/bigip.conf***)* to set the length of time between checks and the length of time to wait for a check response before timeout.

Extended service checks also use data from the **/etc/bigd.conf** file. There are seven ways to use Extended Content Verification and Extended Application Verification to check status. The different checks are listed in Table 6.4:

Keyword in bigd.conf	Usage
ssl	ECV
active	ECV, ECV on nodes w/wildcard ports
reverse	ECV fails check if string is found
external	EAV
gateway	router ping
transparent	transparent node mode
simple	wildcard ports simple check

Table 6.4 *Keywords in* ***/etc/bigd.conf***

Service checking for wildcard servers and ports

The **simple** keyword is necessary to perform simple service checks on nodes with wildcard ports. Use the following syntax to set a check on a node where the check port is not the node port:

```
simple [<node addr>:]<node port> <check port>
```

For example, if a wildcard server is defined with a non-wildcard port:

```
bigpipe vip 0.0.0.0:0 define n1:0
```

To configure the check on it, use the **simple** keyword to designate the wildcard <**server:**><**port**> and <**check port**>:

```
simple n1:0 80
```

Use the following variation on the **active** keyword syntax to configure ECV on nodes with wildcard ports:

```
active <node addr>:0 <check port> [<send string> [<regexp>]]
```

This syntax is only allowed when the node port is 0.

To support EAV on nodes with wildcard ports, an additional variation on the **external** command in the **/etc/bigd.conf** file is added:

```
external <node addr>:0 <check port> [<program name> [<arguments>]]
```

This syntax is only allowed when the node port is 0.

When this syntax is used, the calling convention for the external pinger is changed to:

```
<program name> <node addr> <check port> <arguments>
```

Service checking through transparent nodes

The **/etc/bigd.conf** file supports ECV for transparent nodes. This is done by checking a destination through the particular transparent node you want to check.

The following syntax is supported in the **/etc/bigd.conf** file for ECV through a transparent node:

```
transparent <node_ip>:<port> <site_ip>:<port> [<send_string>
   [<recv_expr>]]
```

The **bigdnode** program uses this syntax to make the appropriate socket option settings for the ECV check.

The following example shows how to set up an ECV check through a transparent node. The following virtual servers are defined for this example:

```
bigpipe vip 0.0.0.0:80 define p1:80 p2:80
```

```
bigpipe vip 0.0.0.0:0  define fw1:0 fw2:0
```

Configure the **/etc/bigd.conf** as shown:

```
transparent p1:80 www.myweb.com:80 'GET /' 'Yahoo'
transparent p2:80 www.myweb.com:80 'GET /' 'Yahoo'
transparent fw1:0 www.myweb.com:80 'GET /' 'Yahoo'
transparent fw2:0 www.myweb.com:80 'GET /' 'Yahoo'
```

Tip

Note that wildcard ports in virtual server definitions no longer require a defined service check port with the node if you do not want port translation. Instead, ***0*** *is used to indicate that port translation should not take place.*

In this example, node **p1:80** is tested by getting the web page **http://www.yahoo.com/**. The web request is routed through **p1**. The transparent node **fw2:0** is tested by getting the same web page (still on port 80), routed through **fw1**.

big3d

The **big3d** daemon answers 3-DNS Controller system queries. 3-DNS uses **big3d** to collect information about the network path between the BIG-IP Controller and the client requesting a connection. The **big3d** utilities calculate performance data, and return the data to the requesting 3-DNS Controller. The 3-DNS Controller uses the path information for its own dynamic load balancing.

You can start or stop the **big3d** process without affecting any other processes on the BIG-IP Controller.

If you no longer want to run the **big3d** process on the BIG-IP Controller, stop the process and remove the corresponding start line from **/etc/rc.local**. The only reason you might want to do this is if your installation once used 3-DNS but no longer uses it.

WARNING

*When the **big3d** agent on the BIG-IP Controller is stopped, the 3-DNS Controller can no longer provide dynamic load balancing for the virtual servers that run on the BIG-IP Controller. This may affect pool definitions on the 3-DNS Controller.*

big3d hardware and software compatibility

The version of the **big3d** daemon on the BIG-IP Controller and the 3-DNS Controller that sends requests to it must be compatible. Any time you upgrade the BIG-IP Controller or the 3-DNS Controller, check to make sure the versions of **big3d** are compatible.

Installing big3d

Run the **big3d** install script on the 3-DNS Controller to install the correct version of **big3d** on the BIG-IP Controller, and add the auto start information to the BIG-IP Controller **/etc/rc.local** file. This

sets up the proper fail-over configuration, so that if the BIG-IP Controller is rebooted or fails over, **big3d** starts automatically on the standby BIG-IP Controller.

Note

*For more information about setting up **big3d**, see the **3-DNS Controller Administrator Guide**.*

Services and port configurations

Communication between the 3-DNS Controller and **big3d** daemon on the BIG-IP Controller depends on the proper management of specific ports.

Outbound iQuery requests

The port used by the iQuery protocol to pass queries and results between the 3-DNS Controller and **big3d** is now registered with the IANA as port 4353.

In previous versions of the BIG-IP Controller, outbound iQuery traffic service used port 245. Current releases of the BIG-IP Controller software and the 3-DNS Controller software enable both of these ports by default, and the **big3d** daemon on the BIG-IP Controller detects iQuery requests on either port.

Firewall ports

The firewall ports 245 and/or 4353 must allow traffic between the BIG-IP Controller and the 3-DNS Controller.

WARNING

Firewalls between the 3-DNS and BIG-IP Controllers must allow traffic on one or both of these ports. If the firewall rejects iQuery traffic, then 3-DNS cannot provide dynamic load balancing for the virtual servers that run on the BIG-IP Controller, which may affect pool definitions on the 3-DNS Controller.

F5 Networks BIG-IP Local Traffic Controller

7

BIG/db Configuration Keys

Supported BIG/db configuration keys

The BIG/db is a database that contains configuration elements for the BIG-IP Controller. Currently, the following configuration options are supported by BIG/db:

- Fail-over
- State mirroring
- Gateway pingers
- Configuration synchronization
- Interface related settings

The BIG/db keys for each of these features are described in the following series of tables.

Using the Failover and Cluster keys

The switch-over daemon (**sod**) uses all **.Failover** keys (Table 7.1). The **sod** and the mirroring daemon (**sfd**) use all **.Cluster** keys. If you change one of these values, you must update the BIG-IP Controller configuration with the **bigpipe failover init** command. This command forces **sod** to reread the BIG/db database and use the new values. To run this command, type the following:

```
bigpipe failover init
```

For more information about **sod**, see *sod*, on page 6-2.

Fail-Over Key Name	Description
Common.Bigip.Failover.AwaitPeerAliveDelay	Delay in seconds before testing whether peer is active. The default value is **2.**
Common.Bigip.Failover.AwaitPeerDeadDelay	Delay in seconds before testing whether the peer has failed. The default value is **1**.
Local.Bigip.Failover.UnitId	This key is required. Each controller must have a unique unit ID of **1** or **2** in the event that network communication is not possible with its peer.

Table 7.1 *The BIG/db keys that store fail-over information*

Fail-Over Key Name	Description
Common.Bigip.Failover.ActiveMode	Use active-active mode if set to **1**. By default, this is off and active/standby mode is used.
Common.Bigip.Failover.ManFailBack	If using active-active mode, the fail-over mechanism waits until receiving a command before surrendering resources to a rebooted machine.
Common.Bigip.Failover.NoSyncTime	By default, one BIG-IP Controller synchronizes its time with the other. Set this key to **1** to turn off the time synchronization feature.
Common.Bigip.Failover.DbgFile	File into which **sod** logs the fail-over debug information.
Common.Bigip.Failover.PrintPeerState	The default value for this key is **0**. Fail-over daemon (**/sbin/sod**) writes the state of its connection to its peer, hardwire and/or network. This information is written to the fail-over daemon's debug log file.
Common.Sys.Failover.Network	Use the network (via the **sfd**) as a backup to, or instead of, the serial line for fail-over if this value is **1**. By default, this feature is off.
Common.Bigip.Cluster.ActiveKeepAliveSec	The default value for this key is **0**. An active unit sends a heartbeat message to its peer with this frequency. Default is **1** second.
Common.Bigip.Cluster.StandbyTimeoutSec	Consider the peer BIG-IP Controller failed if no message is receied within the timeout period. Used by network fail-over. Default is **3** seconds.

Table 7.1 *The BIG/db keys that store fail-over information*

Using StateMirror keys

The mirroring daemon (**sfd**) uses all **StateMirror** keys (Table 7.2). If you change one of these values, you must reboot the BIG-IP Controller to load the new configuration and restart the **sfd**.

State Mirroring Key Name	Description
Common.Bigip.StateMirror.DbgFile	File into which debug information is written, required if debug level is specified (below).
Common.Bigip.StateMirror.DbgLevel	The debug level is composed of the following options: 1 - log reads and writes 2 - log connection attempts and results 4 - log state changes 8 - open log files in append mode, the default is to truncate the files when opening. The debug level is zero by default.
Common.Bigip.StateMirror.NoGC	By default, state mirroring causes mirrored data structures to be deleted when it receives a new connection. This key is brought up to date by the controller's peer. This can cause a delay if the system is absolutely loaded. Turning off the GC is provided as an option.
Common.Bigip.StateMirror.ActiveFile	Enables writing of data from the active unit's kernel into the ActiveFile file. This data file can be read with the **sfread** program.
Common.Bigip.StateMirror.StandbyFile	Enables writing of standby data into the StandBy file. This data file can be read with the **sfread** program.
Common.Bigip.StateMirror.PeerListenPort	Port on which the BIG-IP Controller listens for connections from the active unit. Default is **1028**.
Local.Bigip.StateMirror.Ipaddr	IP address of this BIG-IP Controller.
Local.Bigip.StateMirror.PeerIpaddr	IP address of this BIG-IP Controller's peer controller. A value is required.

Table 7.2 *The BIG/db keys that store state mirroring information*

Using Gateway Pinger keys

The following table lists the Gateway Pinger keys.

Gateway Pinger Key Name	Description
Local.Bigip.GatewayPinger.Ipaddr	IP address or host name of the gateway router for the BIG-IP Controller
Local.Bigip.GatewayPinger.Pinginterval	Ping interval of this BIG-IP Controller gateway pinger
Local.Bigip.GatewayPinger.Timeout	Timeout of the BIG-IP Controller gateway pinger

Table 7.3 *The BIG/db keys that store gateway pinger information*

Using configuration synchronization and Interface keys

The First-Time Boot utility sets the configuration syncronization and **Interface** keys (Table 7.4 and Table 7.5). You should not change these keys manually with **bigdba**.

Configuration Synchronization Key Name	Description
Common.Bigip.ConfigSyncKey	Used for synchronizing the configuration of two BIG-IP Controllers.
Local.Bigip.FTB.DefaultRoute Local.Bigip.FTB.FailoverIp Local.Bigip.FTB.FTP Local.Bigip.FTB.FTP.allow Local.Bigip.FTB.FTP.support Local.Bigip.FTB.HostName = bigip1 Local.Bigip.FTB.HostNumber = 1 Local.Bigip.FTB.Interfaces Local.Bigip.FTB.IsRedundant Local.Bigip.FTB.License Local.Bigip.FTB.Locality = domestic Local.Bigip.FTB.Product = BIG-IP® Local.Bigip.FTB.RootSet Local.Bigip.FTB.RSH Local.Bigip.FTB.RSH.allow Local.Bigip.FTB.RSH.support Local.Bigip.FTB.SupportEmail Local.Bigip.FTB.SupportFax Local.Bigip.FTB.SupportPhone Local.Bigip.FTB.Telnet Local.Bigip.FTB.Telnet.allow Local.Bigip.FTB.Telnet.support = yes Local.Bigip.FTB.TimeZone = US/Pacific Local.Bigip.FTB.Vendor Local.Bigip.FTB.VendorCommonName Local.Bigip.FTB.VendorSupportIp Local.Bigip.FTB.Version = "BIG/ip 3.2" Local.Bigip.FTB.WebServer.Cert.City = Seattle Local.Bigip.FTB.WebServer.Cert.Company = "Company" Local.Bigip.FTB.WebServer.Cert.Country = US Local.Bigip.FTB.WebServer.Cert.Division = undef Local.Bigip.FTB.WebServer.Cert.State = WA Local.Bigip.FTB.WebServer.Prefix = http Local.Bigip.FTB.WebServer.ReplaceKeys = no	First-Time Boot utility fields. These fields are populated by the First-Time Boot utility. These represent a small fraction of the actual fields produced, as the rest are dynamic and dependent upon First-Time Boot utility user input such as host names and interface identifiers.

Table 7.4 *The BIG/db keys that store configuration synchronization information.*

Interface Key Name	Description
Local.Bigip.Interface.*.Name Common.Bigip.Interface.*.Name	The interface name is used as the key for querying or modifying its attributes. The key is common by default, but can be local to accommodate redundant BIG-IP Controllers with different interface names. However, the interface names must all be local or all be common.
Common.Bigip.Interface.*.AddrList.*.Address Common.Bigip.Interface.*.AddrList.*.Netmask Common.Bigip.Interface.*.AddrList.*.Broadcast	Attributes of an interface's addresses. Currently, the BIG/db contains only shared IP addresses.
Common.Bigip.Interface.*.AddrList.*.Type	Type is **Shared** or **True**. **Shared** is associated with an IP alias, **True** is associated with an IP address.
Common.Bigip.Interface.*.AddrList.*.UnitId Common.Bigip.Interface.*.AddrList.*.VlanTag	The unit number with which the shared IP alias is associated.

Table 7.5 *The BIG/db keys that store interface information*

F5 Networks BIG-IP Local Traffic Controller

Index

/etc/bigconf.conf file 3-1
/etc/bigd.conf file 2-7, 3-1, 6-5
adding entries 1-6
and service checking 1-19
changing 1-20
configuring 1-22–1-24
data 1-14, 1-89
/etc/bigd.conf file configuration entries 1-19
/etc/bigip.conf file 1-89, 2-7, 2-53, 2-57, 3-1, 6-5
creating rules in 1-120
defining pools in 1-62
/etc/bigip.license file 3-1
/etc/hosts file 3-1
/etc/hosts.allow file 1-89, 2-7, 3-1
/etc/hosts.deny file 1-89, 2-7
/etc/ipfw.conf file 1-89, 2-7, 3-1
/etc/ipfw.filt file 1-89
/etc/ipfwrate.conf file 1-90, 2-7, 3-1
/etc/ipfwrate.filt file 1-90
/etc/irs.conf file 3-1
/etc/login.conf file 3-1
/etc/netstart file 3-1
/etc/rateclass.conf file 1-90, 2-7, 3-1
/etc/rc file 3-1
/etc/rc.local file 1-100, 3-1, 6-5
/etc/rc.sysctl file 3-1, 5-1–5-23
/etc/services file 1-84, 2-44
/etc/snmpd.conf file 1-90, 2-7, 3-1
/etc/sshd_config file 3-1
/etc/syslog.conf file 3-1
/etc/wideip.conf file 3-1
/usr/contrib/bin/ssh-askpass file 3-2
/var/f5/bigdb/user.db file 3-2
/var/f5/httpd/basicauth/users file 3-2
/var/f5/httpd/conf/cert.conf file 3-2
/var/f5/httpd/conf/httpd.conf file 3-2
/var/f5/httpd/ssl/lib/ssleay.cnf file 3-2
/var/f5/httpd/ssl/lib/ssleay.conf file 3-2
/VENDOR file 3-1
/VERSION file 3-2
-? command 2-4

A

access rates 1-27
active units 1-89
active-active mode
configuring 1-103
configuring shared IP aliases 1-102
running mixed versions of BIG-IP 1-112
synchronizing configuration 1-107
system failover. See system failover
transitioning from active/standby 1-107
updating fail-over daemon 1-107
administration ports 1-40
adminport option 1-39
alias command 1-34–??, 2-5–2-6
application response 1-3
ARP 1-91
ARP requests 1-155, 2-80, 5-24
ASCII characters
and constant operands 1-117

B

backward compatibility 5-3, 5-4
backward-compatible commands 2-87
BIG/db keys
configuration synchronization 7-5
fail-over 7-1
gateway pinger 7-4
interface 7-6
state mirroring 7-3

BIG/db parameters
for active-active mode 1-104, 1-110
BIG-IP Controller types I-7
configuration utilities 4-1
BIG/pipe commands 2-1–2-87
BIG/pipe utility I-2
BIG/top utility
described I-2
big3d utility 6-9
bigd daemon
checking node status 1-2
bigd utility 6-4
bigdnode daemon 1-3
bigdnode program
and service checks 1-7
bigip.bonfire_compatibility_mode variable 5-4
bigip.bonfire_mode variable 5-3
bigip.fastest_max_idle_time variable 5-5
bigip.forwarding_vip_overrides_default_snat 5-8
bigip.halt_reboot_timeout variable 5-9
bigip.improved_fastest 5-10
bigip.max_sticky_entries variable 5-11
bigip.memory_reboot_percent variable 5-12
bigip.open_3dns_lockdown_ports variable 5-13
bigip.open_ftp_ports variable 5-15
bigip.open_rsh_ports variable 5-17
bigip.open_ssh_port variable 5-16
bigip.open_telnet_port variable 5-14
bigip.persist_on_any_port_same_vip variable 5-21
bigip.persist_on_any_vip variable 5-20
bigip.persist_time_used_as_limit variable 5-19
bigip.tcphps_mss_override variable 5-22
bigip.verbose_log_level variable 5-23
bigip.vipnoarp variable 5-24
bigip.webadmin_port variable 5-25
bigpipe commands
and active-active mode 1-111
displaying active data 1-111
mirroring 1-94
bigpipe interface command
for source and destination processing 1-38
bigpipe lb command 1-49
bigpipe ratio command 1-50
bigpipe tping_svc command 1-6
bitmask 1-55, 2-27
broadcast 1-155, 2-80

C

cache server types I-6
cacheable content determination
defined I-6
client IP address variable 1-116
See also client_addr variable
client IP addressses
and rules 1-114
client_addr variable 1-117, 1-118
config utility 4-2
config_failover command 1-88
config_failover utility 4-4
config_rshd utility 4-8
config_sshd utility 4-7
config_telnetd utility 4-6
configsync command 2-7
Configuration 4-4
configuration files 3-1–3-2
configuration keys
BIG/db 7-1–7-6

configuration settings 2-10, 2-57
configuration synchronization key names 7-5
configuration utilities 4-1
Configuration utility
  configuring a pool 1-62
  described I-1
  FTP EAV service check 1-8
  POP3 EAV service check 1-9
  setting up ECV 1-21
Configuration utility requirements I-2
configurations
  clearing memory 2-50
  synchronizing 2-7
  testing files 2-9
  validating 2-9
conn command 2-8
connection limits 1-48
  for node addresses 1-59, 2-32
  for nodes 1-59, 2-32
  for ports 2-44
  for SNAT addresses 2-60
  for virtual servers 1-156, 2-81
connection requests 1-149
connection timeout 2-73
connections 1-161
  client 2-8
  FTP 5-15
  long-lived 1-93
  persistence 2-35
  Telnet 5-14
  verifying 1-19
  virtual server 1-154, 2-80
constant operand types 1-117
content retrieval I-6
content types
  for caching I-6
controller performance 1-35
controller synchronization 1-88, 1-89
controllers
  active and standby 1-93
  setting preferred active unit 1-100
cookie names 1-120
cookies 1-29

D

-d command 2-9
default routers 1-32, 1-91
destination address affinity 1-75
destination IP addresses
  and ECV service checks 1-20
  as filter criteria 1-25
  changing 1-35
  for creating filters 1-29
  translating 1-35–1-37
destination ports
  and ECV service checks 1-20
destination processing
  and interface function 1-35, 1-36
  and IP packets 1-37
dynamic load balancing modes 1-46–1-48

E

EAV pingers 1-7
EAV service checkers
  implementing 1-3
EAV service checks
  and FTP services 1-7
  and NNTP services 1-10
  and POP3 service 1-8
  and SMTP services 1-9
  and SQL-based services 1-11
  described 1-2, 1-3
  installing 1-5
  setting up 1-3
EAV. See service checks
ECV 1-13
  normal 1-15
  regular expressions 1-16

reverse 1-15
SSL 1-15

ECV configuration information 1-20

ECV service checks
and transparent nodes 1-19, 1-21

ECV. See service checks

encrypted connections I-8

expressions
listed 1-118
parts of 1-117
See also logical expressions

extended application verification. See service checks

Extended Content Verification (ECV). See ECV

extended content verification. See service checks

external interfaces 1-53, 1-155, 2-14, 2-27, 2-80

external network. See external interfaces

external service checker types 1-2

external service checkers
installing 1-5
internal behavior of 1-2
location of 1-4, 1-7
requirements 1-4
sample 1-5
starting 1-6
use of 1-2

F

-f command 2-10

F5 Configuration utility 4-4, 5-1

failover command 2-11

fail-over process 1-93

fail-safe
features of 1-96

fail-safe settings 1-87–1-92

Fastest mode 1-47

filter types 1-25, 1-27

filters
defined 1-25

First-Time Boot utility 4-2
defined I-1

forwarding virtual servers
configuring 1-32

FQDNs 1-119

F-Secure SSH client option
and encrypted communications I-8
as a remote shell I-2

FTP
on ports 1-85

FTP pinger arguments 1-7

FTP protocol
and service checking 1-2

FTP servers
verifying connections to 1-19

FTP service
and service checks 1-7

Fully Qualified Domain Names
See FQDNs

G

gateway command 2-12

gateway fail-safe 1-96–1-98

gateway fail-safe messages 1-98

gateway pinger key names 7-4

global load balancing
See also load balancing

H

-h command 2-13

hard-wired fail-over 1-98

Hash mode 1-73

header_tag_string 1-120
-help command 2-13
help command 2-4
HTTP headers
configuring rules 1-114
HTTP request data 1-114
and pool selection 1-114
HTTP request headers
and content caching I-6
HTTP request string variables
and header data 1-119
and rules 1-115
as variable operands 1-118
replacing 1-118
HTTP request variable names 1-119
http_cookie variables 1-120
http_header variables 1-120
http_host variables 1-119
http_method variables 1-119
http_uri variables 1-119
http_version variables 1-119
httpd server 4-5

I

if statements 1-116
Insert mode 1-70
intelligent cache population
defined I-6
interface cards 1-91
interface command 1-44, 2-14–2-20
interface key names 7-6
interface properties 1-35
interface security
configuring 1-40
interface types 1-35
internal interfaces 1-53, 2-14
internal network. See internal interfaces
intervals
for external service checks 1-6
IP address notation 1-117
IP addresses 2-21, 2-27
and ECV service checks 1-20
changing 1-37, 2-16
defining I-1
destination 1-149
for clients 1-77
See also client IP addresses
IP aliases
configuring additional 1-102
IP forwarding 5-26
setting up 1-30–1-32
system control variables 1-31
IP packet filters
and destination IP addresses 1-26
and source IP addresses 1-26
configuring 1-26
in the F5 Configuration utility 1-26
IP packets
and rule evaluation 1-117
ipalias command 2-21

L

-l command 2-22
last hop feature 1-37, 2-16
lb command 2-23
Least Connections mode 1-47
lists
See also member lists
See node lists
literals. See constant operands
load balancing
and ECV service checks 1-13
and priority levels 1-48
configuring I-1
monitoring I-1

onset 1-115
transparent nodes 1-149
using node lists 1-131

load balancing modes
described 1-46–1-48
Priority 2-51
Ratio 2-51
setting 2-23
setting global 1-48

load balancing pool selection
and client IP address 1-115
and HTTP request data 1-114

load balancing pools 1-62, 2-36, 2-79
and client address variables 1-116
defining 1-62
referencing by servers and rules 1-114, 1-122
selecting via use statements 1-116
sharing 1-132

load balancing rules 2-79

lockdown keyword 1-39

logging 5-23

logical expressions
listed 1-119

logical operators
listed 1-119

M

MAC addresses 1-42, 2-19

mail servers
and service checks 1-9
verifying connections to 1-19

maint command 2-25

masked dot notation
and constant operands 1-117

media access control. See MAC addresses

messages
gateway fail-safe 1-98

MIB I-2

mirroring
described 1-93–1-96
global 1-94

mirroring commands 1-94

mirroring. See persistence 1-157, 2-83

monitoring methods
and command-line utilities I-2

N

naming scheme
for PID file 1-4

NAT addresses
translating 1-36

nat command 1-57, 2-27–2-29

NATs
defining 1-55

net.inet.ip.forwarding variable 5-26

net.inet.ip.sourcecheck variable 5-11, 5-27

netmask 1-55, 1-155, 2-27, 2-80

network address translation (NAT). See nat command

network address translations. See NATs

network interface cards. See NIC

network traffic 1-91
and HTTP requests 1-115
and IP packet filters 1-25
and ITC (intelligent traffic control) 1-29
identifying 1-29

network virtual server 1-153

network-based failover 1-98

newsgroups
and service checks 1-10

NIC 1-44, 2-14–2-20

NICs (Network Interface Cards)
viewing settings for 1-38

NNTP service
and service checks 1-2, 1-10

NNTP_pinger arguments 1-10
node addresses 1-30, 1-53, 1-54
and priority levels 1-49–1-51
and ratio weights 1-49–1-51
connection limits 1-59, 2-32
ratio weights 2-51
node aliases 2-5
node command 1-58–1-61, 2-31–2-34
node configuration 1-32, 1-53
node lists
and load balancing methods 1-131
example 1-131
using 1-131
node ping 1-140–1-141, 2-5, 2-64–2-65, 2-68, 6-6
node ping timer 1-139
node pings 1-2
for node status 1-2
node ports
and ECV service checks 1-20
node status
checking 1-2
nodes
and load balancing 1-47–1-48
connection limits 1-59, 2-32
connections 2-27
service checks 2-66–2-69
virtual server mappings 2-78
non-transparent cache servers
described I-6

O

Observed mode
defined 1-47
open keyword 1-39
operands 1-117
operators 1-117, 1-119
operators. See relational operators

P

packets
forwarding and rejecting 1-25
Passive mode 1-72
performance
See controller performance
persist command 2-35
persistence 1-81, 5-19, 5-20, 5-21
advanced options 1-66–1-81
and mirroring 1-93–1-99
conditions for 1-78
for connections 1-161
for virtual ports 2-35
maintaining across virtual servers 1-81
See also destination address affinity
See also device persistence
See also HTTP cookie persistence
See also simple persistence
See also SSL persistence
simple 1-68
SSL 1-67
using persist mask 1-77
pid file 1-3, 1-4
ping. See node ping
pingnode. See node ping
pool command 1-62–1-64, 2-36–2-42
pool selection 1-114, 1-115
pools
persistence 1-67
pools. See load balancing pools
POP3 protocol
and service checking 1-2
POP3 service
and service checks 1-8
POP3_pinger arguments 1-8
port command 1-86, 2-44–2-45
port frequencies 1-21
port timeouts 1-21

ports 6-10
access to 1-84
defined 2-44
displaying numerically 2-30
idle connection timeout 2-71
persistence 2-35
RSH 5-17
service checking 1-144–1-145
SSH 5-16
translation properties 1-157, 2-82
udp 2-73
wildcard 1-150–1-153

predicates 1-117

Predictive mode
defined 1-47

priority levels 1-49

Priority mode 1-48, 2-51

procedures
configuring gateway fail-safe 1-96
configuring IP aliases 1-103
configuring NATs 1-55
configuring network fail-over 1-99
configuring persistence for virtual servers 1-68–1-69
configuring SNAT address mappings 1-135
configuring SNAT global properties 1-134
configuring SSL persistence 1-67
defining virtual server mappings 1-151
enabling active-active mode 1-105
returning to active/standby mode 1-113
setting idle connections 1-142
setting IP forwarding 1-31
setting node ping timer 1-139
setting service check timer 1-143
setting up ECV service checking 1-18
synchronizing redundant systems 1-107
turning off port translation 1-152

R

-r command 2-50

rate classes 1-27, 1-28

rate filters 1-27, 1-28

rates of access 1-27

ratio command 2-51–2-52

Ratio mode 1-46, 1-47, 1-51, 2-51

ratio weights 2-51, 2-52
setting 1-49

rc.sysctl file 2-7

receive expressions 1-17

reconfig-httpd utility 4-5

redundant system modes 1-101, 1-113

redundant systems 1-87–1-92
active unit 1-89
advanced features of 1-87, 1-113
configuring gateway fail-safe 2-12
configuring hop pools 1-157, 2-82
default mode 1-101
displaying unit number 2-75
failover 2-11
fail-safe interfaces 1-92
gateway fail-safe. See gateway fail-safe
mirroring 1-137, 2-60
See also active/standby mode
See also mirroring
See also network-based failover
sharing MAC addresses 1-43, 2-19
standby unit 1-89
synchronizing 2-7

regular expression strings
and constant operands 1-117

relation expressions
using HTTP request variables in 1-119

relational operators
listed 1-118

remote shell. See RSH

Rewrite mode 1-71

root password 1-89
defining I-1
round robin 1-45
routing table 1-32
RSH 5-17
rule behavior 1-114
rule command 2-53–2-55
rule elements
listed and described 1-122
rule evaluation 1-117
rule examples 1-114, 1-127
AOL rule 1-129
cache content rule 1-129
client IP address and IP protocol 1-115
cookie rule 1-127
IP protocol specific rule 1-131
language rule 1-128
rule statements 1-116
rules
and client address variable 1-116
configuring 1-122
creating 1-120
defined 2-53
elements 1-122
example 1-115
listing of 2-55
load balancing 2-79
load balancing pools 2-53
referencing multiple pools 1-132
referencing pools 1-122
selecting pools 1-114–1-120

S

secure connections I-8
secure network address translation (SNAT). See snat command
secure network address translation (SNAT). See SNATs
secure shell 4-7, 5-16
send strings 1-16
server resource allocation 1-29
service checks 2-66–2-67, 2-69
and bigd daemon 1-2
configuring 1-13–1-24
customizing 1-2
EAV 1-143, 2-66, 2-69, 6-6–6-8
ECV 1-13, 1-15, 1-143, 2-66, 2-69, 6-6–6-8
simple 1-143, 2-66, 2-69, 6-6–6-7
timeouts 2-67
tping_svc 2-69
troubleshooting 1-12
service ping internals
setting 1-6
service response 1-3
services 1-84
FTP configuration 4-6
telnet configuration 4-6
simple keyword 1-144
simple persistence
SMTP protocol
and service checking 1-2
SMTP service 1-9
SMTP_pinger arguments 1-9
SNAT addresses
translating 1-36
snat command 1-137, 2-58–2-61
SNAT connections
mirroring 1-95
SNATs
address mappings 1-135
connection limits 1-133
defining 1-133–1-136
global properties 1-133
TCP idle connection timeout 1-133
UDP idle connection timeout 1-134
SNMP MIB I-2
sod utility 1-100, 6-2

source address translation
and packet processing 1-37, 2-16
source checking 5-27
source IP addresses 1-54
SQL Enterprise Manager 1-12
SQL language
and service checking 1-2
SQL-based service checks
troubleshooting 1-12
SQL-based services
and service checks 1-11
SSH 5-16
SSH client option
See F-Secure SSH client option
SSL connections I-8
SSL persistence 1-33
standby mode 1-100
standby units 1-89
state mirroring 1-112
state mirroring key names 7-3
statements
for rules 1-116
function of 1-116
sticky entries 5-11
string variables 1-119
subjects 1-117
summary command 2-62
switch-over daemon 6-2
SYN packets 1-115
synchronization. See controller synchronization
sysctl command 5-2
system control variables 5-1–5-23
system fail-over
disabling automatic fail back 1-108
in active-active mode 1-108, 1-110
placing back in service 1-109
placing in standby mode 1-109
system setup I-1
system utilities
big3d 6-9–6-10
bigd 6-4–6-8
sod 6-2–6-3

T

target IP addresses
See destination IP addresses
TCP
setting idle connection timers 1-142
traffic 1-141
TCP handshakes
proxying 1-115
TCP SYN packets 1-115
test accounts
creating 1-12
text editor 5-1
timeout_node command 1-141, 2-64–2-65
timeout_svc command 2-66–2-67
timer settings
configuring 1-145
idle connections 1-141–1-143
node ping 1-139–1-140
service check 1-143–1-144
tping_node command 2-68
tping_svc command 2-69–2-70
translation properties 1-157, 2-82
transparent network devices 1-147, 1-149
transparent node mode 5-3, 5-4
transparent nodes 1-149, 1-150
and service checking 1-19
verifying function of 1-19, 1-21
treaper command 2-71–2-72

U

UDP

setting idle connection timers 1-142
traffic 1-141

udp command 2-73–2-74

unit command 2-75

URLs
and ECV service checks 1-20
and http_uri variable 1-119

usage statistics 2-62

use statements 1-116

utilities I-2
BIG/pipe commands 2-1–2-87
configuration 4-1
F5 Configuration 4-4
First-Time Boot 4-2
system 6-2–6-10

V

-v command 2-76

variable names
string variable names 1-119

variable operands
and rules 1-117
defined 1-117
types 1-117

variables 1-117

vip command 1-160, 2-78–2-85

virtual addresses
defining a netmask 1-155, 2-80
displaying information 1-159, 2-84
statistics 1-159, 2-84
translation properties 1-157, 2-82

virtual ports. See ports

virtual server configuration
referencing rules 1-121
See also rules
using node lists 1-131

virtual server mappings 1-78, 1-80, 2-78
defining standard 1-148–1-149
defining wildcard 1-151–1-153

virtual servers 1-81, 1-160, 2-78–2-85
additional features 1-161
and persistence 1-78
configuring 1-153
configuring addresses 1-105
configuring hop pools 1-157, 2-82
configuring load balancing pools 2-79
defining standard 1-147, 1-149
defining wildcard 1-147, 1-149–1-153
displaying unit number 2-75
enabling 1-158, 2-83
forwarding 1-54
mirroring 1-95
persistence 1-68
ports 2-78
rules 2-53, 2-79
sharing pools and rules 1-132

VLAN communication 1-44, 2-20

W

web administration 5-25

web applications
verifying 1-2

web server access I-1

web server content
verifying 1-19

web servers 1-14

wildcard virtual servers
and ECV service checks 1-21