

3 DNS

Administrator Guide

3DNS® Controller Administrator Guide

version 2.0

Service and Support Information

Product Version

This manual applies to version 2.0 of the 3DNS® Controller platform.

Obtaining Technical Support

Web	tech.f5.com
Phone	(206) 505-0888
Fax	(206) 505-0802
Email (support issues)	support@f5.com
Email (suggestions)	feedback@f5.com

Contacting F5 Networks

Web	www.f5.com
Toll-free phone	(888) 88BIG-IP
Corporate phone	(206) 505-0800
Fax	(206) 505-0801
Email	sales@f5.com
Mailing Address	200 1st Avenue West Suite 500 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© 2000 by
F5 Networks, Inc.
Seattle, Washington
All rights reserved. Printed in U.S.A.

F20

Trademarks

F5, BIG/ip, and 3DNS are registered trademarks of F5 Networks, Inc. see/IT and global/SITE are trademarks of F5 Networks, Inc. Other product and company names are registered trademarks or trademarks of their respective holders.

Export Regulation Notice

The 3DNS® Controller, if originally sold within the United States, may include cryptographic software. Under the Export Administration Act, the United States government may consider it a criminal offense to export this 3DNS® 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.

Canada Compliance

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

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.

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 Thomas Boutell (boutell@boutell.com).

This product includes software developed byJef Poskanzer (jef@acme.com).

This product includes software developed by Eric Young (eay@cryptosoft.com).

The material included in Appendix C was taken from the Internet Software Consortium's web site. The ISC is a non-profit group, and their web address is <http://www.isc.org/>.

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 (for example, BIG/ip V1.5 to BIG/ipV1.6). 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 (for example, BIG/ip V1.x to BIG/ip V2.0). 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 for the 3DNS® Controller, whether such software is provided separately or as an integral part of a hardware product. As used herein, the term “Software” will refer to all such software, and the corrections, updates, new releases and new versions of such software. A product that consists of Software only will be referred to as a “Software Product” and a combination Software/Hardware product will be referred to as a “Combination Product.” All Software is licensed, not sold, by F5. 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 solely on a single central processing unit owned or leased by Licensee. 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. Licensee may make one back-up copy of any Software Product, provided the back-up copy contains the same copyright and proprietary information notices as the original Software Product. Licensee is not authorized to copy the Software contained in

---

a Combination Product. The Software incorporates certain third party software which is used subject to licenses from the respective owners.

3. Restrictions. The Software, documentation, and the associated copyrights are owned by F5 or its licensors, and are protected by law and international treaties. Except as provided above, 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 laws and governmental 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 exporting or using cryptographic software.

5. Limited Warranty.

a) 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.

b) 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 F5 product that fails during the warranty

---

period at no cost to Licensee. Any 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 Licensee, at F5's expense, no later than 7 days after receipt by F5. Title to any returned product 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.

c) Restrictions. The foregoing limited warranties extend only to the original Licensee, and do not apply if a Software Product or Combination Product (i) has been altered, except by F5, (ii) has not been installed, operated, repaired, or maintained in accordance with F5's instructions, (iii) has been subjected to abnormal physical or electrical stress, misuse, negligence, or accident or (iv) 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. 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 THE SOFTWARE, 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 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. 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 THE SOFTWARE OR OTHER GOODS OR SERVICES FURNISHED TO LICENSEE BY F5, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination. This License is effective until terminated, and will automatically terminate if Licensee fails to comply with any of its provisions. Upon termination of this License, the Licensee will destroy the Software and documentation and all copies or portions thereof.
8. 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.



Table of Contents

Chapter 1

Introduction to the 3DNS Controller

Welcome to the 3DNS Controller	1-1
3DNS Controller specifications	1-1
Internet protocol and network management support	1-2
Security features	1-2
Configuration scalability	1-3
Configuration and monitoring tools	1-3
Browser support	1-4
System synchronization options	1-5
Configurable data collection for server status and network path data ..	1-5
Redundant system configurations	1-6
IP packet filtering	1-6
Load balancing modes	1-7
Managing traffic on a global network	1-9
A sample network layout	1-9
Synchronizing configuration information and broadcasting performance metrics	1-11
Using a 3DNS Controller as a standard DNS server	1-11
Load balancing connections across the network	1-13
Working with BIG/ip Controllers and other products	1-14
What's new in version 2.0	1-16
New configuration and monitoring tools	1-16
Redundant system options	1-16
Multiple network interface cards	1-17
Data center definitions	1-17
Enhanced configuration and metrics synchronization	1-17
Production rules	1-18
Enhanced dynamic load balancing	1-18
Include statements for the wideip.conf configuration file	1-19
Finding help and technical support resources	1-19

Chapter 2

Preparing for Installation

Reviewing the installation tasks	2-1
Understanding the installation phases	2-1
Working with configuration tools	2-2
Planning issues for the hardware setup	2-3
Gathering basic information	2-4
Addressing special hardware configuration issues	2-4
Planning issues for the network setup	2-7

Defining data centers and servers	2-8
Setting up data collection with the big3d agent	2-9
Setting up SNMP probing for hosts	2-14
Planning sync groups	2-15
Setting up communications between 3DNS Controllers, BIG/ip Controllers, and big3d agents	2-16
Planning issues for the load balancing configuration	2-25
Understanding the wideip key	2-25
Choosing a load balancing mode	2-26
Using static modes	2-28
Using dynamic load balancing modes	2-31
Ensuring availability for e-commerce, FTP, and other services that use multiple ports	2-35
Using the LDNS round robin wide IP attribute	2-36
Using advanced traffic control features	2-36
Configuring topology-based access control	2-37
Setting up IP packet filtering	2-37
Defining production rules	2-38
Planning DNS zone file management	2-38
Replacing your DNS servers with 3DNS Controllers as master DNS servers for your domain	2-40
Running 3DNS Controllers as DNS masters only for wide IP sub-domains	2-41

Chapter 3

Setting up the Hardware

Unpacking and installing the hardware	3-1
Reviewing the hardware requirements	3-1
Familiarizing yourself with the 3DNS Controller hardware	3-3
Environmental requirements and usage guidelines	3-4
Installing and connecting the hardware	3-6
Running the First-Time Boot utility	3-8
Gathering the information	3-9
Starting the First-Time Boot utility	3-10
Defining a root password	3-10
Defining a host name	3-11
Configuring a default route	3-11
Configuring a time zone	3-11
Configuring the interfaces	3-11
Configuring remote administration	3-15
Configuring settings for the 3DNS web server	3-16
Identifying additional controllers in the network	3-17
Configuring the NameSurfer application for zone file management ..	3-18

Confirming your configuration settings	3-18
Committing your configuration settings to the system	3-19
Enabling remote login tools	3-20
Preparing workstations for command line access	3-20
Downloading the F-Secure SSH client from the 3DNS web server ..	3-21
Downloading the F-Secure SSH client using FTP	3-21
Setting up the F-Secure SSH client on a Windows 95 or Windows NT workstation	3-23
Setting up the F-Secure SSH client on a UNIX workstation	3-24

Chapter 4

Defining the Network Setup

Setting up a basic configuration	4-1
Setting up a data center	4-2
Setting up servers	4-5
Defining 3DNS Controller servers	4-6
Defining BIG/ip Controller servers	4-9
Defining host servers	4-14
Configuring host SNMP settings	4-17
Configuring SNMP agents on hosts	4-20
Setting up sync groups	4-27
Setting the time tolerance value	4-29
Configuring global variables	4-29
Configuring IP filters	4-31
Defining the filter criteria	4-31
Configuring Sendmail	4-33
Finding the mail exchanger for your domain	4-33
Setting up Sendmail	4-34

Chapter 5

Configuring Basic Load Balancing

Getting started with load balancing	5-1
Defining a wide IP	5-2
Understanding pools	5-3
Adding a wide IP	5-3
Troubleshooting manual configuration problems	5-8
Changing global variables that affect load balancing	5-9
Setting global alternate and fallback modes	5-10
Understanding TTL and timer values	5-11

Chapter 6

Configuring Specialized Load Balancing

Configuring specialized load balancing using advanced modes	6-1
Setting up Quality of Service (QOS) mode	6-1
Understanding QOS coefficients	6-2
Example: Emphasizing completion rate	6-3
Customizing the QOS equation	6-3
Using the Dynamic Ratio option	6-6
Setting up Global Availability mode	6-8
A Global Availability configuration example	6-9
Setting up load balancing for services that require multiple ports	6-10
An example configuration for e-commerce services	6-12
Setting up topology-based features	6-13
Setting up Topology-based access control	6-13
An example configuration for topology access control	6-19
Using the topology load balancing mode	6-22

Chapter 7

Monitoring and Administration

Monitoring and administration utilities provided on the 3DNS Controller ..	7-1
Working with the 3DNS Maintenance menu	7-1
Configuring wide IPs	7-3
Viewing statistics	7-3
Working with the big3d agent	7-4
Managing synchronized files	7-6
Working with security issues	7-6
Using the 3DNS web server	7-7
Working with syncd	7-8
Controlling network traffic patterns with production rules	7-9
Setting up production rules in the F5 Configuration utility	7-9
Working with the production rules scripting language	7-15
Inserting production rules in the wideip.conf file	7-16
Execution and management of production rules	7-16
The if statement	7-17
The when statement	7-19
The every statement	7-20
Production rule actions	7-20
Production rule examples	7-21
Setting up the 3DNS SNMP agent	7-26
Downloading the MIBs	7-26
Understanding configuration file requirements	7-27

Configuring the 3DNS SNMP agent settings	7-30
Configuring options for the checktrap script	7-32
Changing passwords for the 3DNS Controller	7-33
Changing passwords and adding new user IDs for the 3DNS web server	7-33
Viewing system statistics	7-35
Viewing summary statistics	7-35
Viewing current global values	7-42
Viewing data center statistics	7-42
Viewing sync group statistics	7-43
Viewing wide IP statistics	7-44
Viewing 3DNS Controller statistics	7-46
Viewing BIG/ip Controller statistics	7-46
Viewing prober statistics	7-48
Viewing host statistics	7-50
Viewing virtual server statistics	7-52
Viewing path statistics	7-54
Viewing local DNS statistics	7-55

Appendix A

Wideip.conf Syntax

Overview of the wideip.conf file	A-1
Using include files	A-2
Syntax for include files	A-2
Statements	A-4
Syntax rules	A-5
The globals statement	A-6
The datacenter statement	A-25
The sync_group statement	A-27
The server statement	A-28
The wide IP statement	A-39
The topology statement	A-46
Comments	A-48
Sample 3DNS Controller configuration file	A-50
Sample 3DNS Controller configuration file using include files	A-64
Understanding cur_ values	A-78
How cur_ values are used	A-79

Appendix B

3DNS Controller Utilities and Scripts

Utilities	B-1
3dparse	B-1
sod-named	B-3
3ndc	B-4
named	B-5
Configuring syslog for 3DNS messages	B-7
syslogd	B-12
log2mail	B-13
big3d	B-14
bigdb	B-15
bigdba	B-17
bind2namesurfer	B-21
checktrap	B-22
config_httpd	B-23
config_httpsd	B-24
config_namesurfer	B-25
config_rshd	B-25
config_sshd	B-26
syncd	B-26
Scripts	B-31
3dns_admin_start	B-31
3dns_auth	B-31
3dns_dump	B-33
3dns_sync_metrics	B-33
3dns_web_config	B-33
3dns_web_passwd	B-33
3dnsmaint	B-34
3dprint	B-34
3ndc	B-35
big3d_check	B-35
big3d_install	B-35
big3d_restart	B-36
big3d_version	B-36
edit_lock	B-37
edit_wideip	B-37
install_key and F5makekey	B-37
syncd_checkpoint	B-38
syncd_rollback	B-39
syncd_start	B-40
syncd_stop	B-41

Appendix C

BIND 8 Information

BIND 8 overview	C-1
Statements	C-1
acl statement	C-2
key statement	C-3
logging statement	C-4
options statement	C-5
server statement	C-6
zone statement	C-7
Comments	C-8
Converting older configuration files to BIND 8 format	C-10
Relating BIND information to 3DNS Controller wide IP definitions	C-10
Before defining a wide IP	C-10
Defining a wide IP	C-12
Understanding zone minimums	C-20
Replacing your DNS servers with 3DNS Controllers as master DNS servers for your domain	C-21
Running 3DNS Controllers as DNS masters only for wide IP sub-domains	C-23

Appendix D

DNS Resource Records

Overview	D-1
Types of resource records	D-2
Common types	D-2
Other types	D-5

Glossary

Index



1

Introduction to the 3DNS Controller

- **Welcome to the 3DNS Controller**
- **3DNS Controller specifications**
- **Managing traffic on a global network**
- **What's new in version 2.0**
- **Finding help and technical support resources**

Welcome to the 3DNS Controller

Welcome to the *3DNS® Controller Administrator Guide*. This guide describes how to set up the 3DNS Controller hardware and how to set up your network and load balancing configurations, as well as other 3DNS Controller features. The Administrator guide also includes the software specifications for the 3DNS Controller platform, and it offers some sample configurations that can help you in planning your own configuration.

3DNS Controller specifications

The 3DNS Controller is a network appliance that manages and balances traffic over global networks. The 3DNS Controller manages network traffic patterns using load balancing algorithms, topology-based routing, and production rules that control and distribute traffic according to specific policies. The system is highly configurable, and its web-based and command line configuration utilities allow for easy system set up and monitoring.

The 3DNS Controller provides a variety of features that meet special customer needs including:

- ❖ E-commerce sites that need to guarantee the availability of multiple ports
- ❖ Internationally distributed sites that prefer to restrict local clients to local servers
- ❖ Production rules that change the load balancing configuration according to current traffic patterns or time of day
- ❖ SNMP monitoring of the 3DNS Controller
- ❖ Customizable load balancing modes

Internet protocol and network management support

The 3DNS Controller supports both standard DNS protocol and the F5 iQuery protocol (a protocol used for collecting dynamic load balancing information). The 3DNS Controller also supports administrative protocols, such as Simple Network Management Protocol (SNMP), and Simple Mail Transfer Protocol (SMTP) (outbound only), for performance monitoring and notification of system events. For administrative purposes, you can use the F-Secure SSH client (distributed only in the US), which provides a secure shell connection, rsh, Telnet, and FTP. The F5 Configuration utility supports secure connections via SSL (distributed only in the US), as well as standard HTTP connections.

The 3DNS Controller's SNMP agent allows you to monitor status and current traffic flow using popular network management tools, including the F5 Configuration utility. The SNMP agent provides detailed data such as current connections being handled for each virtual server.

Security features

The 3DNS Controller offers a variety of security features that can help prevent hostile attacks on your site or equipment.

❖ **Secure administrative connections**

3DNS Controllers distributed in the US support secure shell administrative connections via F-Secure SSH. The 3DNS web server, which hosts the web-based F5 Configuration utility, supports SSL connections as well as user authentication.

❖ **Secure iQuery communications**

3DNS Controllers distributed in the US also support Blowfish encryption for iQuery communications between controllers running the **big3d** agent.

❖ **TCP wrappers**

TCP wrappers provide an extra layer of security for network connections.

❖ **IP address filtering**

The IP filtering feature, based on BSD IP packet filtering, specifically accepts or denies connections received from particular IP addresses or ranges of IP addresses.

Configuration scalability

The 3DNS Controller is a highly scalable and versatile solution. You can configure the 3DNS Controller to manage up to several hundred domain names, including full support of domain name aliases. The 3DNS Controller supports a variety of media options, including Fast Ethernet, Gigabit Ethernet, and FDDI, and also supports multiple network interface cards that can provide redundant or alternate paths to the network.

Configuration and monitoring tools

The 3DNS Controller provides the following web-based and command line administrative tools that make for easy set up and configuration.

First-Time Boot utility

The First-Time Boot utility is a wizard that walks you through the initial system set up. The utility helps you quickly define basic system settings, such as a root password and the IP addresses for the interfaces that connect the 3DNS Controller to the network. The First-Time Boot utility also helps you configure access to the 3DNS web server, which hosts the web-based F5 Configuration utility, as well as the NameSurfer application that you can use for DNS zone file management.

F5 Configuration utility

The F5 Configuration utility is a web-based application that you use to configure and monitor the 3DNS Controller. Using the F5 Configuration utility, you can define the load balancing configuration, along with the network set up, including data

centers, sync groups, and servers used for load balancing and path probing. In addition, you can configure advanced features such as topology settings, IP filters, and the SNMP agent. The F5 Configuration utility also monitors network traffic, current connections, load balancing statistics, and the operating system itself.

The 3DNS web server, which hosts the F5 Configuration utility, provides convenient access to downloads such as the SNMP MIB and documentation for third-party applications such as NameSurfer.

NameSurfer application

The NameSurfer™ application is a third-party application, produced by Data Fellows, that automatically configures DNS zone files associated with domains handled by the 3DNS Controller. You can use NameSurfer to configure and maintain additional DNS zone files on 3DNS Controllers that run as master DNS servers. The F5 Configuration utility provides direct access to the NameSurfer application, as well as the corresponding documentation for the application.

3DNS Maintenance menu

The 3DNS Maintenance menu is a command line utility that executes scripts which assist you in configuration and administrative tasks, such as installing the latest version of the **big3d agent** on all your systems, or editing the load balancing configuration files. You can use the 3DNS Maintenance menu directly on the 3DNS Controller, or you can use the menu when connected to the controller via a remote shell, such as the SSH client (US only), or a standard **rsh** client.

Browser support

The F5 Configuration utility, which provides web-based access to the 3DNS Controller system configuration and features, supports the following browser versions:

- ❖ Netscape Navigator 4.5 or later
- ❖ Microsoft Internet Explorer, version 4.01 or later

System synchronization options

The 3DNS Controller sync group feature allows you to automatically synchronize configurations from one 3DNS Controller to the other 3DNS Controllers in the network, allowing for simplified administrative management. The synchronization feature offers a high degree of administrative control. For example, you can set the controller to synchronize a specific configuration file set, and you can also set which 3DNS Controllers in the network receive the synchronized information and which ones do not.

Configurable data collection for server status and network path data

The 3DNS Controller platform includes a **big3d** agent, which is an integral part of 3DNS Controller load balancing. The **big3d** agent continually monitors the availability of the servers that the 3DNS Controller load balances. It also monitors the integrity of the network paths between the servers that host the domain and the various client local DNS servers looking to connect to the domain. The **big3d** agent runs on 3DNS Controllers and BIG/ip Controllers distributed in various locations in your network. Each **big3d** agent broadcasts its collected data to all of the 3DNS Controllers in your network, ensuring that all 3DNS Controllers work with the latest information.

The **big3d** agent offers a variety of configuration options that allow you to choose the types of data collection methods you want to use. For example, you can configure the **big3d** agent to track the number of hops along a given network path, and you can also set the **big3d** agent to collect host server performance information using the SNMP protocol.

Redundant system configurations

A **redundant system** is essentially a pair of 3DNS Controller units, one operating as an active unit responding to DNS queries, and one operating as a standby unit. If the active unit fails, the standby unit takes over and begins to respond to DNS queries while the other controller reboots and becomes a standby unit.

The 3DNS Controller actually supports two methods of checking status of the peer system:

❖ **Hardware-based fail-over**

In a system set up with hardware-based fail-over, the two units in the system are connected to each other directly using a fail-over cable attached to the serial ports. The standby controller checks on the status of the active controller every second using this serial link. The controllers check on each other's status using that link.

❖ **Network-based fail-over**

In a system set up with network-based fail-over, the two units in the system communicate with each other across an Ethernet network instead of going across a dedicated fail-over serial cable. The standby Controller checks on the status of the active controller every second using the Ethernet. The controllers check each other's status using that link.

◆ **Note**

In a network-based fail-over configuration, the standby 3DNS Controller immediately takes over if the active unit fails. If a client had queried the failed controller, and not received an answer, it automatically re-issues the request (after 5 seconds) and the standby unit, functioning as the active controller, responds.

IP packet filtering

The 3DNS Controller supports easy configuration of the BSD operating system method of IP packet filtering. In the F5 Configuration utility, you can configure individual IP packet filters, which can control both in-bound and out-bound network traffic. For example, you can specify a single IP address, or a range of IP

addresses, from which the 3DNS Controller either accepts or denies network traffic. You can also specify one or more IP addresses to which you specifically want to allow or prevent out-bound connections.

Load balancing modes

The 3DNS Controller offers several different load balancing modes, including static modes that base load balancing on a pre-defined distribution pattern, and dynamic modes that base load balancing on current network information such as the round trip time between a requesting client and a web server.

Static load balancing

Static load balancing distributes connections based on pre-defined distribution patterns, and does not take current server or network performance into account. The 3DNS Controller supports the following static load balancing modes:

- ❖ **Round Robin**

Round Robin mode is a basic load balancing mode that distributes connections evenly across all servers, passing each new connection to the next server in line.

- ❖ **Ratio**

The Ratio mode distributes new connections across servers in proportion to a user-defined ratio. The distribution of replies is weighted Round Robin. For example, if one server runs on a high-speed server and two other servers run on older machines, you could set the ratio so that the high-speed server receives twice as many connections as either of the two older servers.

- ❖ **Global Availability**

Global Availability mode distributes connections to a list of servers, always sending a connection to the first available server in the list.

- ❖ **Random**

Random mode distributes connections in a random pattern.

❖ **Topology**

Topology allows you to direct or restrict traffic flow by entering network information into the configuration file. This allows you to develop proximity-based mapping. For example, customers in a particular geographic region can be sent to servers within that same region. The 3DNS Controller determines the proximity of servers by comparing the client's LDNS IP address to the IP address of the available servers.

Dynamic load balancing

Dynamic load balancing bases connection distribution on current server and network performance information gathered by the **big3d** agent. The different dynamic load balancing modes incorporate different performance factors.

❖ **Quality of Service**

The Quality of Service (QOS) load balancing mode takes a variety of performance factors into account. You can configure the QOS mode to rate different performance factors higher or lower than others, or you can configure the QOS mode to treat all factors as being equally important. The quality of the service equation calculates a performance score based on the following factors:

- Total round trip time between the server and the client LDNS
- Total number of network hops between the server and the client LDNS
- Number of packets currently processed
- Percentage of packets completed
- Topological distribution

❖ **Round Trip Times**

Round Trip Times mode sends each new connection to the server that demonstrates the best round trip time between the server and the client LDNS.

❖ **Hops**

Hops mode sends each new connection to the server that has the fewest number of intermediate systems transitions between the server and the client LDNS.

❖ **Packet Rate**

Packet Rate mode sends each new connection to the server that has the least amount of network traffic.

❖ **Completion Rate**

Completion Rate mode sends each new connection to the server that has the fewest number of dropped packets.

❖ **Least Connections**

Least Connections mode sends each new connection to the node that currently hosts the fewest current connections. Note that you can use Least Connections mode only to load balance servers managed by BIG/ip Controllers.

Managing traffic on a global network

This section provides a brief overview of how 3DNS Controllers work within a global network and how they interact with other BIG/ip Controllers and host machines in the network. The section also illustrates how the 3DNS Controller works with the **big3d** agents that run in various locations in the network, as well as the local DNS servers that make DNS requests on behalf of clients connecting to the Internet.

The following sample configuration shows 3DNS Controllers that load balance connections for a sample Internet domain named **domain.com**.

A sample network layout

3DNS Controllers sit in specific data centers in your network and they work in conjunction with BIG/ip Controllers and with generic host servers that also sit in your network data centers. All 3DNS Controllers in the network can receive and respond to DNS resolution requests from the local DNS servers that clients use to connect to the domain.

Figure 1.1 illustrates the layout of the 3DNS Controllers, BIG/ip Controllers, and host servers in the three data centers. The Los Angeles data center houses one 3DNS Controller and one BIG/ip Controller, as does the New York data center. The Tokyo data center houses only one 3DNS Controller and one host server.

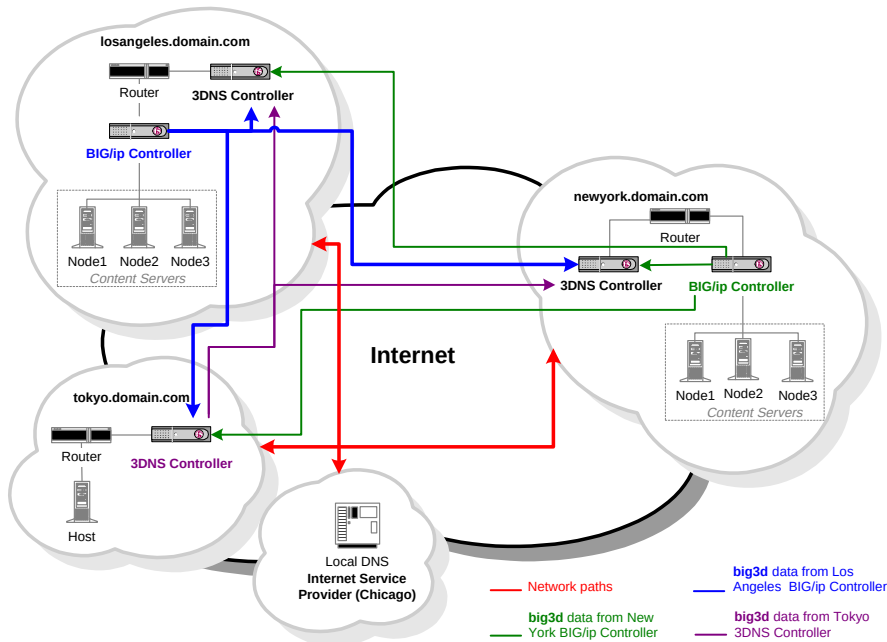


Figure 1.1 A sample network layout

In the Los Angeles and New York data centers, the **big3d** agent runs on the BIG/ip Controller, but in the Tokyo data center, the **big3d** agent runs on the 3DNS Controller. Each **big3d** agent collects information about the network path between the data center where it is running and the client's local DNS server in Chicago, as illustrated by the red lines. Each **big3d** agent also broadcasts the

network path information it collects to the 3DNS Controllers running in each data center, as illustrated by the green, blue, and purple lines.

◆ Note

*All BIG/ip Controllers and 3DNS Controllers in a data center typically run a **big3d** agent.*

Synchronizing configuration information and broadcasting performance metrics

3DNS Controllers typically work in sync groups where a group of controllers shares load balancing configuration settings. In a sync group, any controller that has new configuration changes can broadcast the changes to any other controller in the sync group, allowing for easy administrative maintenance. To distribute metrics data among the controllers in a sync group, the **principal 3DNS Controller** sends requests to the **big3d** agents in the network, asking them to collect specific performance and path data. Once the **big3d** agents collect the data, they each broadcast the collected data to all controllers in the network, again allowing for simple and reliable metrics distribution.

Using a 3DNS Controller as a standard DNS server

When a client requests a DNS resolution for a domain name, DNS sends the request to the 3DNS Controller that is authoritative for the zone (running as a master DNS server for the domain). The 3DNS Controller chooses the best available virtual server out of a pool, and then returns a standard DNS answer record (an **A** record) to the requesting local DNS server. The local DNS server uses the answer for the period of time defined within the **A** record. Once the answer expires, however, the local DNS server must request name resolution all over again to get a fresh answer.

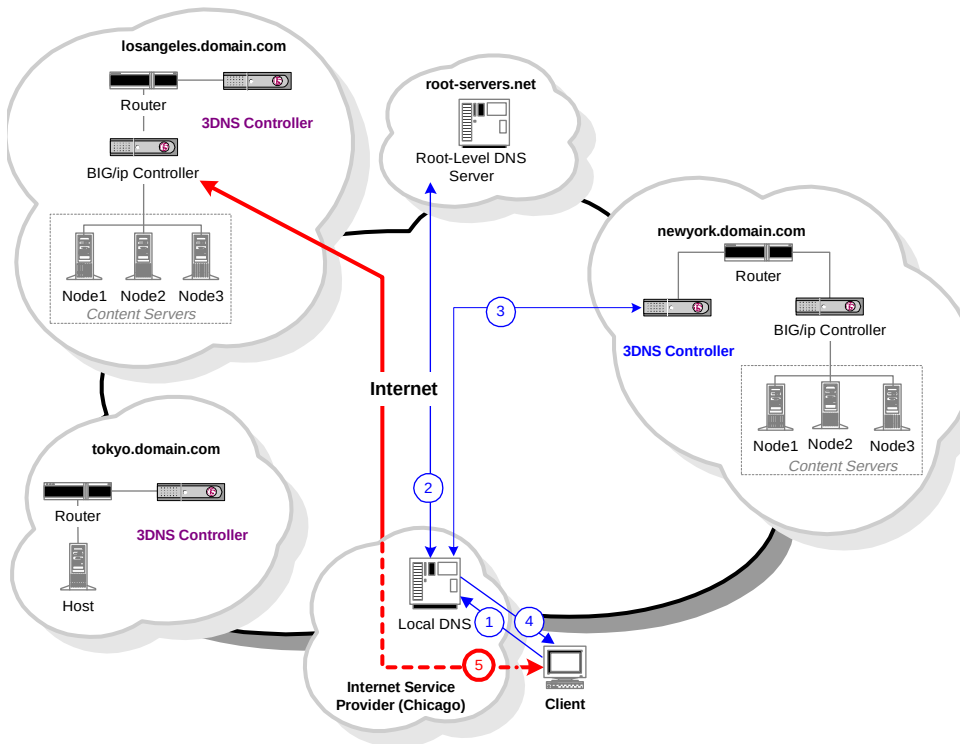


Figure 1.2 Name resolution process

Figure 1.2 illustrates the specific steps in the name resolution process.

1. The client connects to an Internet Service Provider (ISP) and queries the local DNS to resolve the domain name **www.domain.com**.
2. If the information is not already in the local DNS server's cache, the local DNS server queries a root server (such as InterNIC's root servers). The root server returns the IP address of a DNS associated with **www.domain.com**, which in this case runs on the 3DNS Controller.

3. The local DNS then connects to the 3DNS Controller looking to resolve the **www.domain.com** name. The 3DNS Controller uses a load balancing mode to choose an appropriate server to receive the connection, and returns the server's IP address to the local DNS.
4. The local DNS ends the connection to the 3DNS Controller and passes the IP address to the client.
5. The client connects to the IP address via the ISP.

◆ **Note**

The dotted portion of line 5 indicates that the actual hardware for this step is not shown, due to the number of ways ISPs can configure their networks. The actual machines that handle all other transaction events are shown, so all other lines are solid.

Load balancing connections across the network

Each of the 3DNS Controller load balancing modes can provide efficient load balancing for any network configuration. The 3DNS Controller bases load balancing on pools of virtual servers. When a client requests a DNS resolution, the 3DNS Controller uses the specified load balancing mode to choose a virtual server from a pool of virtual servers. The resulting answer to this resolution request is returned as a standard **A** record.

Although some load balancing configurations can get complex, most load balancing configurations are relatively simple, whether you use a static load balancing mode or a dynamic load balancing mode. More advanced configurations can incorporate multiple pools, as well as advanced traffic control features, such as topology or production rules. (For a list of individual load balancing modes, see *Load balancing modes*, on page 1-7).

Working with BIG/ip Controllers and other products

The 3DNS Controller balances connections across a group of virtual servers that run in different data centers throughout the network. You can manage virtual servers from the following types of products:

- ❖ **BIG/ip Controllers**

A BIG/ip Controller virtual server maps to a series of content servers.

- ❖ **Generic hosts**

A host virtual server can be an IP address or an IP alias that hosts the content.

- ❖ **Other load balancing products**

Other load balancing products map virtual servers to a series of content hosts.

Figure 1.3 illustrates the hierarchy of virtual server management in our sample configuration.

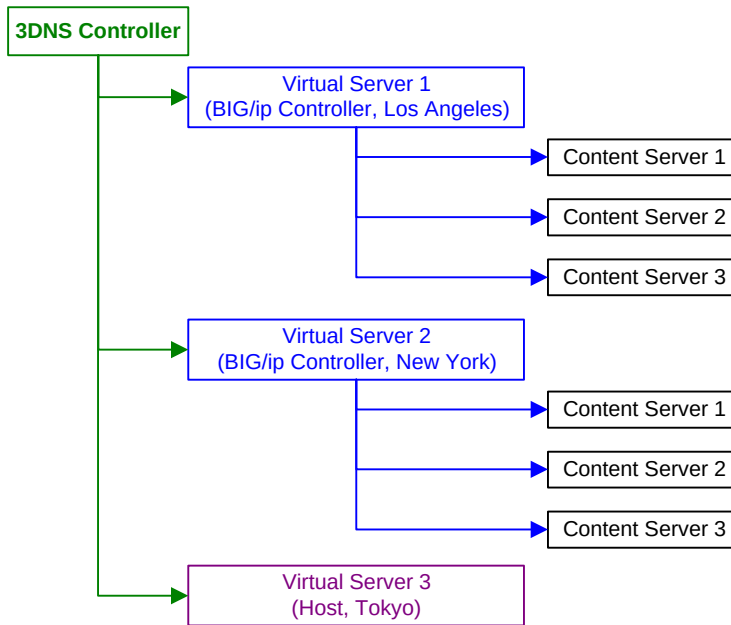


Figure 1.3 Load balancing management

How 3DNS Controller differs from BIG/ip Controller

While both controllers provide load balancing, one of the significant differences between the 3DNS Controller and the BIG/ip Controller is that the 3DNS Controller responds to DNS requests issued by an LDNS on behalf of a client, while the BIG/ip Controller provides connection management between the client and the back-end server.

Once the 3DNS Controller returns a DNS answer to an LDNS, the conversation between the LDNS and the 3DNS Controller ends, and the client connects to the IP address returned by the 3DNS Controller. Unlike 3DNS, the BIG/ip Controller sits between the client and the content servers. It manages the client's entire conversation with the content server.

What's new in version 2.0

The 3DNS Controller offers the following major new features in version 2.0.

New configuration and monitoring tools

The 3DNS Controller now supports the following configuration and monitoring tools:

- ❖ **The F5 Configuration utility**

The F5 Configuration utility is a web-based application that provides easy configuration of data centers, sync groups, wide IPs, and all other 3DNS Controller features and settings. The F5 Configuration utility also provides enhanced system monitoring similar to that found in the old Web Administration tool found in previous releases of the 3DNS Controller.

- ❖ **NameSurfer**

The NameSurfer application provides easy configuration and maintenance for DNS zone files on 3DNS Controllers that run as master DNS servers.

- ❖ **The SNMP MIB**

The 3DNS Controller includes a new proprietary SNMP MIB that you can use in conjunction with SNMP-based network management applications to monitor 3DNS Controller performance and system state.

Redundant system options

The 3DNS Controller now offers two types of redundant system configurations:

- ❖ **Hardware fail-over**

Hardware fail-over is the standard fail-over configuration where two 3DNS Controller units in the system are connected directly by a fail-over cable. This provides the highest level of reliability, because it does not depend on any network equipment to get the important fail-over data from one unit to the other.

❖ **Network fail-over**

Network fail-over is a configuration option that allows you to set up two individual 3DNS Controllers as a redundant system, without having a direct hard-wired connection between the two units. Instead, the units transfer the fail-over data via the network. This option works well in many situations, but does not provide as much reliability as the hardware fail-over setup. You may want to consider using this option to provide an additional layer of fail-over redundancy in a system that is currently configured for hardware fail-over.

Multiple network interface cards

All 3DNS Controller products now support two network interface cards. Use of the second network interface card is optional, and it can provide you an additional layer of redundancy. The separate network interface cards can connect through different routers or gateways to the same network, allowing for more than one available network path.

Data center definitions

The new data center feature allows you to map out the network layout of the 3DNS Controllers, BIG/ip Controllers, and host machines that you use for load balancing. One major benefit of the data center feature is that you can use the **big3d** agent on one controller in the data center to collect network path data on behalf of all controllers and hosts that run in the same data center.

Enhanced configuration and metrics synchronization

The 3DNS Controller now supports **sync groups**, which you can use to define one or more groups of controllers that share configuration settings and path statistics. A sync group contains a **principal 3DNS Controller** that broadcasts its configuration settings and path statistics at set intervals to the remaining 3DNS Controllers, referred to as **receiver 3DNS Controllers**, in the sync

group. This can make system administration across a global network much easier, because it requires you to make configuration changes to only one controller, instead of all controllers running in the network.

Note that the **big3d** agent now uses broadcasting to distribute current path statistics and server status to all 3DNS Controllers running in the network. The principal controller in a sync group issues requests to the **big3d** agent, and once the **big3d** agent retrieves the requested information, it sends the information not only to the principal 3DNS Controller, but to all controllers. When the principal controller in a sync group broadcasts its configuration setting and path statistics to the receiver controllers, it includes the path statistics only as a backup to the path statistics that the controller should receive from the broadcasting **big3d** agent.

Production rules

You can use the production rules feature to dynamically change the load balancing configuration depending on current network traffic patterns, or time of day. The F5 Configuration utility provides easy configuration for production rules, and the sample **wideip.conf** file provided in *Appendix A* includes two examples of the production rule language.

Enhanced dynamic load balancing

The 3DNS Controller now supports the following enhancements for dynamic load balancing:

- ❖ **Hops load balancing mode**

The Hops load balancing mode is based on the traceroute utility, and it distributes connections based on the fewest number of hops required between the server and the client.

- ❖ **New options for the Quality of Service load balancing mode**

The Quality of Service load balancing mode now supports a **hops coefficient**, and a new attribute referred to as **dynamic ratio**. The hops coefficient simply adds a **hops** factor to the **qos** equation, where it takes into account the number of network hops between

each server and the client. The dynamic ratio attribute allows the 3DNS Controller to use **qos** scores as ratios. When you apply the dynamic ratio attribute, the 3DNS Controller distributes connections across all available virtual servers in proportion to each server's **qos** score. Servers that receive higher **qos** scores receive a larger percentage of the connection load than servers which receive lower **qos** scores.

❖ **LDNS round robin attribute**

When you apply the LDNS round robin attribute, the 3DNS Controller returns a full list of available servers for name resolution. Certain types of browsers cache the list of available servers and use the list for subsequent connections, rather than returning to the 3DNS Controller for a fresh name resolution each time.

❖ **SNMP probing for hosts**

The 3DNS Controller now supports a new probing method used to collect performance information for hosts. You can use SNMP probing for any host that runs a UCD, Solstice, or an NT SNMP MIB.

Include statements for the wideip.conf configuration file

The **wideip.conf** file now supports include statements. The new default structure of the **wideip.conf** file includes only the load balancing configuration settings. The metrics data, which was previously stored in the file, is now stored separately from the **wideip.conf** file and included only by reference.

Finding help and technical support resources

You can find additional technical documentation about the 3DNS Controller in the following locations:

❖ **Release notes**

The release note for the current version of the 3DNS Controller is available from the home page of the F5 Configuration utility.

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 3DNS Controller features**

You can find help online in three different locations:

- The F5 Configuration utility home page has a PDF version of this administrator guide. Note that some 3DNS Controller upgrades replace the online administrator guide with an updated version of the guide.
- The F5 Configuration utility also has online help for each screen. Simply click the Help button in the toolbar.
- Individual 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 3DNS Controller displays the syntax and usage associated with the command.

❖ **Third-party documentation for software add-ons**

The F5 Configuration utility contains online documentation for all third-party software included with the 3DNS Controller, including NameSurfer and 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, and updates for administrator guides (in PDF format). To access this site, you need to obtain a customer ID and a password from the F5 Help Desk.



2

Preparing for Installation

- **Reviewing the installation tasks**
- **Planning issues for the hardware setup**
- **Planning issues for the network setup**
- **Planning issues for the load balancing configuration**
- **Using advanced traffic control features**
- **Planning DNS zone file management**

Reviewing the installation tasks

This chapter covers the planning issues that you need to address before you install and set up the 3DNS Controller. It explains the three different phases on an installation, along with load balancing configuration options and important DNS planning issues. If you are planning to do a simple configuration, briefly review the hardware setup and network setup requirements, along with the load balancing options. If you are planning to do a more advanced configuration, we recommend that you read about each advanced feature you want to work with in Chapter 6 before you begin the installation.

Be sure to review the DNS zone file management section included later in this chapter. It provides an important overview of the process involved in setting up a 3DNS Controller that runs as a master DNS server, as well as integrating 3DNS Controllers into networks that have an existing master DNS server.

Understanding the installation phases

You can set up a basic 3DNS Controller configuration, or an advanced 3DNS Controller configuration. Regardless of whether you are creating a basic or an advanced configuration, you always go through three installation phases:

❖ **Hardware setup**

The hardware setup phase simply gets the hardware up and running, and connected to the network. This includes completing the configuration settings that allow remote access for command line access via a UNIX shell, and for the 3DNS web server which hosts the F5 Configuration utility. You need to do the hardware setup on all 3DNS Controllers in your network.

❖ **Network setup**

During the network setup phase, you define the network layout of the servers that the 3DNS Controller uses for load balancing and path probing. You define the different data centers in the network, as well as the equipment that runs in each data center. You also set up a **sync group**, which is a group of 3DNS Controllers that share system configurations. Note that at this

point, you can define the configuration on the 3DNS Controller that is the principal controller in the sync group, and then all other controllers in the sync group automatically copy the configuration from the principal controller.

❖ **Load balancing configuration**

The load balancing configuration is the last phase of a 3DNS Controller installation. This phase can be simple and fast if you are setting up a very basic configuration, particularly if you allow the NameSurfer application to handle the DNS zone file management for you. If you are planning to use a more advanced load balancing configuration, however, you may need to do some additional planning.

DNS zone file integration happens at different times, depending on how you decide to handle DNS zone file management, and depending on whether you are installing new 3DNS Controllers in your network, or upgrading existing 3DNS Controllers in your network. Issues relating to DNS zone file management are covered in detail in *Planning DNS zone file management*, on page 2-38.

Working with configuration tools

When you configure the 3DNS Controller, you have two configuration tool options:

- ❖ We recommend that you use the browser-based Configuration utility in conjunction with the NameSurfer application. Using these tools together provides two significant benefits that make for simplified system administration. First, when you make configuration changes to any 3DNS Controller in the network, the sync feature automatically broadcasts the changes to all other 3DNS Controllers in the network. Second, when you add or modify wide IP definitions in the Configuration utility, the NameSurfer application automatically makes the appropriate updates to the DNS zone files. You do not need to do any DNS zone file maintenance yourself.
- ❖ If you prefer, you can do the configuration manually using the **Edit 3DNS Configuration** command on the 3DNS Maintenance menu (a command line tool). However, this method requires that

you make corresponding changes to DNS zone files manually, which can be quite complex. It also requires that you individually configure each 3DNS Controller in the network.

If you have worked with previous versions of the 3DNS Controller, you can view statement syntax in the Configuration utility. As you define various aspects of the product, use the **View Conf** button to display the existing configuration.

 WARNING

We do not recommend that you switch between using the Configuration utility, and manually configuring the 3DNS Controller. If you manually configure a 3DNS Controller and then attempt to use the Configuration utility to modify the configuration, you risk losing custom configuration settings not supported by the Configuration utility.

Planning issues for the hardware setup

In the hardware setup phase, you connect the 3DNS Controller hardware to the network, and you run the First-Time Boot utility on each of the 3DNS Controllers in the network. The First-Time Boot utility is a command line wizard that helps you define basic required system settings such as the IP address, host name, root user ID, and other information necessary for the controller and its web server to be accessible over the network.

When you run the First-Time Boot utility, you must work on a keyboard and monitor, or a serial terminal, connected directly to the 3DNS Controller. However, after you complete the hardware setup phase, you can finish the remaining installation tasks from a remote administrative workstation.

Gathering basic information

The prompts in the First-Time Boot utility are generally self-explanatory, and you should be able to answer them as long as you prepare the following information before you start running the utility:

- ❖ Root administration password for each controller
- ❖ IP addresses for each network interface, and a shared IP alias for each redundant system
- ❖ Host names used on each network interface
- ❖ IP addresses of remote workstations that need to access the controllers
- ❖ Time zones for each controller
- ❖ IP addresses and remote connection requirements (**ssh** versus **rsh**) for all 3DNS Controllers and BIG/ip Controllers in the network configuration
- ❖ Authentication certificate settings (company name, address, etc.) for 3DNS web servers that run on US 3DNS Controllers

Addressing special hardware configuration issues

Before you start the hardware setup, you may want to review the following items which address configuration and management issues for redundant systems, systems that use more than one network interface, and DNS zone file management.

Are you setting up a stand-alone unit or a redundant system?

If you are setting up a stand-alone unit, you need one IP address and host name for each of the interfaces you plan to connect to the network. If you are setting up a redundant system, you need one IP address for each network interface card in each unit, as well as a shared IP alias for the primary network interface, and a shared IP alias for the secondary network interface (if you are connecting the redundant system to more than one network).

If you are setting up a redundant system, are you using hardware-based fail-over or network-based fail-over?

Hardware-based fail-over is a redundant system that connects two 3DNS Controller units directly to each other using a fail-over serial cable. **Network-based fail-over** is a redundant system where two units are either connected to each other directly using an Ethernet cable, or they are connected indirectly via an Ethernet network. Of the two units in a redundant system, one runs as the **active unit**, managing all DNS resolution requests, and the other runs as the **standby unit**, waiting to take over in case the active unit fails and reboots. The communication between the units, such as fail-over notification, runs across either the fail-over cable in the hardware-based redundant system, or the network in the network-based redundant system.

When you run the First-Time Boot utility, it prompts you to enter the IP address of the other unit in the system.

What events trigger a fail-over in a redundant system?

The 3DNS Controller tracks two key aspects of the system to validate system performance. In a redundant system, there are two events that indicate a system failure, and trigger a fail-over.

- ❖ If the **named** daemon becomes unresponsive, or if you manually stop the daemon using the **ndc stop** or **ndc restart** commands, the 3DNS Controller treats this as system failure and initiates a fail-over.
- ❖ If the 3DNS Controller fails to detect any traffic on its network interfaces, it attempts to create traffic to test the integrity of the interface. If the test fails, the 3DNS Controller treats this as a system failure and initiates a fail-over.

How do redundant systems work with the sync group feature?

If you include a redundant system in a sync group, you include the system by specifying the system's shared IP alias. When a controller in the sync group broadcasts new information to the redundant system, only the active unit receives the updated information. The moment a fail-over occurs and the standby unit

becomes active, the next synchronization check compares timestamps on the configuration files and immediately updates the unit with the current system configuration, path metrics, and DNS zone files.

If you have a sync group that includes a redundant system, you may want to make test changes to the configuration on the standby unit, without having those changes synchronized to the other controllers in the group. As long as the unit runs as a standby, the changes remain local. Once you validate your changes and you are ready to broadcast them to the other controllers in the group, you can run the **bigpipe fo** command on the active unit to force a fail-over. After the system fails over to the standby unit with your new configuration changes, those changes broadcast to the other controllers in the group during the next synchronization pass.

Note that you always run the risk of having the active unit initiate a fail-over to the standby unit before you can verify that the configuration changes are complete. Although the 3DNS Controller does not commit any configuration changes that violate syntax rules, there is always the possibility that a fail-over may occur before you complete your configuration changes.

Are you planning on using more than one network interface?

The First-Time Boot utility prompts you to configure the primary network interface, and then asks if you want to configure more interfaces, or if you want to skip to the next section of the utility. If you want to configure another network interface, you simply enter the same type of information you entered for the first interface. The other interfaces can connect to a separate network, or they can act as a redundant paths to the same network that the first interface is connected to.

Do you want to set up automatic DNS zone file management?

The First-Time Boot utility asks you if you want to use the NameSurfer application as the master for DNS zone files. We recommend that you always run NameSurfer as the master for DNS zone files. When you define or modify wide IPs in the F5 Configuration utility, NameSurfer automatically makes the

corresponding changes to the DNS zone files. The NameSurfer application also provides you with easy management of high-level domain zone files unrelated to the wide IP configuration.

If you are planning on transferring existing BIND files from a master DNS server to the 3DNS Controller, you do not configure NameSurfer when you run the First-Time Boot utility; you configure the application later on in the installation process. For more details about this and other DNS zone file management issues, see *Planning DNS zone file management*, on page 2-38.

Planning issues for the network setup

After you finish running the First-Time Boot utility and get each controller connected to the network, you can do the network setup and load balancing configuration on one controller, and let the sync group feature automatically broadcast the configuration to the other controllers in the network. You do not have to configure 3DNS Controllers individually, unless you are planning an advanced configuration that requires different configurations for different data centers, or you are doing the configuration manually.

During the network setup phase, you define three basic aspects of the network layout, in the following order:

- ❖ **Data centers**

Data centers are the physical locations that house the equipment you use for load balancing.

- ❖ **Servers**

The servers you define in the network setup include only the 3DNS Controllers, BIG/ip Controllers, and host machines that you use for load balancing.

❖ **Sync group**

A **sync group** defines the group of 3DNS Controllers that shares configuration settings and path data.

◆ **Note**

During the network setup phase of configuration, we recommend that you connect to the 3DNS Controller from a remote workstation where you can do the remaining configuration tasks using the web-based F5 Configuration utility.

Defining data centers and servers

It is important that you define all of your data centers before you begin defining servers. When you define a server, you specify the data center where the server runs by choosing a data center from the list of data centers you have already defined.

To define a data center, you need only specify the data center name. To define a server, however, you need to specify the following items:

- ❖ Server type (3DNS Controller, BIG/ip Controller, or host)
- ❖ Server IP address (or shared IP alias for redundant systems)
- ❖ Name of the data center where the server runs
- ❖ **big3d** agent settings (BIG/ip Controllers and 3DNS Controllers only)
- ❖ Virtual servers managed by the server (BIG/ip Controllers and hosts only)
- ❖ SNMP host probing settings (hosts only)

The most important part of planning data centers and servers is to decide how to set up the **big3d** agent, and which ports you need to open for communications between the controllers in your network. The following sections in this chapter, *Setting up data collection with the big3d agent* and *Setting up communications between 3DNS Controllers, BIG/ip Controllers, and big3d agents*, provide help with determining how both of these issues affect your installation.

Understanding how the time tolerance variable affects sync groups

The time tolerance value is a global variable that defines the number of seconds that one 3DNS Controller's time setting is allowed to be out of sync with another 3DNS Controller's time setting. If the difference between the times on the controllers is higher than the time tolerance, the time setting on the controller running behind is reset to match the controller with the most recent time. For example, if the time tolerance is 5 seconds, and one 3DNS Controller is running 10 seconds ahead of the other, the controller running behind has its time reset to match the one running 10 seconds ahead. If the second controller was running only 2 seconds ahead of the other, the time settings would remain unchanged. The values are 0, 5, and higher (values of 1-4 are automatically set to 5, and 0 turns off time syncing). The default setting is **10** seconds.

The time setting on 3DNS Controllers is important because a 3DNS Controller compares time stamps on files when deciding whether to synchronize files with other 3DNS Controllers in the sync group.

Setting up data collection with the big3d agent

The **big3d** agent collects performance information on behalf of the 3DNS Controller. The **big3d** agent runs on both 3DNS Controllers and BIG/ip Controllers. The default setting is to run a **big3d** agent on all controllers in the network, but you can turn off the **big3d** agent on any controller at any time.

Setting up the **big3d** agents involves the following tasks:

❖ Installing big3d agents on BIG/ip Controllers

Each time you receive new 3DNS Controller software, the software includes the latest version of the **big3d** agent. You need to distribute that copy of the **big3d** agent to the BIG/ip Controllers in the network. Use the **3dnsmaint** menu item to automatically install the appropriate version of the **big3d** agent for each version of the BIG/ip Controller. See the release notes provided with the 3DNS Controller software for information about which BIG/ip Controller versions the current **big3d** agent supports.

❖ **Specifying which factories a specific big3d agent manages**

When you define BIG/ip Controller and 3DNS Controller servers, you can change the default **big3d** agent settings on a specific controller. You can change the number of factories the **big3d** agent runs and turn specific factories on and off.

❖ **Setting up communications between big3d agents and controllers**

Before the **big3d** agents can communicate with the 3DNS Controllers in the network, you need to configure the appropriate ports and tools to allow communication between the BIG/ip Controllers and 3DNS Controllers in the network. These planning issues are discussed in *Setting up communications between 3DNS Controllers, BIG/ip Controllers, and big3d agents*, on page 2-16.

Path data and server performance

A **big3d** agent collects the following types of performance information used for load balancing. This information is broadcast to all 3DNS Controllers in your network.

❖ **Virtual server availability**

The **big3d** agent queries virtual servers to verify whether they are up and available to receive connections. For name resolution, the 3DNS Controller uses only those virtual servers that are **up**.

❖ **Network path round trip time**

The **big3d** agent calculates the round trip time for the network path between the data center and the client's local DNS server making the resolution request. Round trip time is used in determining the best virtual server when utilizing the Round Trip Times or the Quality of Service modes.

❖ **Network path packet loss**

The **big3d** agent calculates the packet completion percentage for the network path between the data center and the client's local DNS server making the resolution request. Packet completion is used in determining the best virtual server when utilizing the Completion Rate or the Quality of Service modes.

❖ Hops along the network path

The **big3d** agent calculates the number of intermediate systems between the data center and the client's local DNS server. Hops are used in determining the best virtual server when utilizing the Hops or the Quality of Service modes.

❖ Server performance

The **big3d** agent calculates the packet rate of the BIG/ip Controller or SNMP-enabled hosts. Packet rate is used in determining the best virtual server when utilizing the Packet Rate or the Quality of Service modes

❖ BIG/ip virtual server performance

The **big3d** agent calculates the number of connections to a virtual server defined on a BIG/ip Controller. The number of connections is used to determine the best virtual server when utilizing the Least Connections mode.

Installing the big3d agent on BIG/ip Controllers

To install the **big3d** agent on the BIG/ip Controllers in your network, log on to the 3DNS Controller using either a remote shell, or the serial terminal or keyboard and monitor attached directly to the controller. At the command prompt, type **3dnsmaint** to open the 3DNS Maintenance menu and choose the **Install and Start big3d** command.

Understanding factories run by big3d agents

To gather performance information, the **big3d** agent uses different types of factories. A factory is a process which collects different types of data. The **big3d** agent currently supports four factory types:

❖ Probing factory

A probing factory collects several types of information using ICMP, TCP, or UDP. This factory queries host virtual servers and LDNSs. Host virtual servers are checked to determine their **up** or **down** state. For LDNSs, the probing factory uses the response to calculate the round trip time and packet loss between the LDNS and the data center.

❖ **Hops factory**

A hops factory uses the traceroute method to calculate the number of hops along the network path between a specific data center and a client LDNS.

❖ **SNMP factory**

An SNMP factory uses conversations with SNMP agents that run on host servers to collect performance metrics for the host.

❖ **Discovery factory**

A discovery factory acts as a backup to a probing factory, and the **big3d** agent runs a discovery factory only when a probing factory fails to get a response from a specific LDNS. The **big3d** agent uses the discovery factory to look for an alternate port on an LDNS that can respond to the queries issued by a probing factory. If the discovery factory finds an open port, it returns the port number to the 3DNS Controller, which stores the number to use for future path probe attempts.

❖ **Permanent factories**

Two permanent factories collect performance information. One factory collects information from the BIG/ip Controller when it exists, the other collects the number of packets being processed per second. These factories are not configurable.

The standard configuration specifies that each BIG/ip Controller and 3DNS Controller in the network runs a **big3d** agent using five prober factories, no SNMP factories, one discovery factory, no hops factories, and the two default factories. In the BIG/ip Controller or 3DNS Controller server definition, you can change the number of factories that the **big3d** agent runs. For example, the default number of hops factories is set to **0**; if you want to run a hops factory, you change the setting to **1** or more.

Understanding the data collection and broadcasting sequence

The **big3d** agents collect and broadcast information on demand. The principal 3DNS Controller in the sync group issues a data collection request to all **big3d** agents running in the network. In turn, the **big3d** agents collect the requested data using its factories, and then broadcasts that data to all 3DNS Controllers running in the network, including the principal controller that issued the request.

Important notes about tracking LDNS probe states

The 3DNS Controller tracks the state of path data collection for each LDNS that has ever requested a name resolution from the controller. Table 2.1 shows the six states that can be assigned to an LDNS. Note that you can view the state of LDNS servers in the statistics area of the F5 Configuration utility.

State	Description
Needs Probe	The big3d agent has never collected data for the LDNS, or the data has expired.
Idle	The big3d agent successfully collected data for the LDNS, and is waiting for the next collection request.
In Probe	The big3d agent is currently collecting data for the LDNS.
Needs Discovery	The big3d agent failed to collect data for the LDNS using its standard protocols and ports, and now needs to run the LDNS through a discovery factory.
In Discovery	The big3d agent is currently running the LDNS through a discovery factory to look for an alternate available port.
Suspended	The big3d agent failed to discover a port open for data collection, and the LDNS is no longer eligible for data collection requests.

Table 2.1 Probe and discovery states for individual client LDNS servers

Evaluating big3d agent configuration trade-offs

You must run a **big3d** agent on each BIG/ip and 3DNS Controller. If you are using dynamic load balancing modes, you **must** have a **big3d** agent running on at least one controller in each data center in order to gather the necessary path metrics.

The load on the **big3d** agents depends on two factors: the time-to-live (**ttl**) settings that you assign to the different types of data the agents collect, and the number of factories that each agent runs. The shorter the **ttls**, the more frequently the agents need to refresh the data. While short **ttls** guarantee that you always have valid data readily available for load balancing, they also increase the

frequency of data collection. The more factories a **big3d** agent runs, the more metrics it can refresh at one time, and the more quickly it can refresh data for the 3DNS Controller.

Another factor that can affect data collection is the number of client LDNS servers that make name resolution requests. The more LDNS servers that make resolution requests, the more paths that the **big3d** agent has to collect. While round trip time for a given path may vary constantly due to current network load, the number of hops along a network path between a data center and a specific LDNS does not often change. Consequently, you may want to set short **ttls** for round trip time data so that it refreshes more often, but set high **ttls** for hops data because it does not need to be refreshed often.

Setting up SNMP probing for hosts

The host probing feature uses SNMP conversations to collect performance data for the host. The 3DNS Control uses the performance data for dynamic load balancing modes such as Packet Rate and Quality of Service. The **big3d** agent actually uses a special SNMP factory to collect metrics from host servers. The SNMP factory establishes a conversation with an SNMP agent running on a given host server. From the SNMP conversation, the factory determines the following types of information about the host:

- ❖ Memory utilization
- ❖ CPU utilization
- ❖ Disk space utilization
- ❖ Bytes in/out
- ❖ Packet rate

The Configuration utility displays the host metrics in the Host statistics screen. The 3DNS Controller bases the dynamic load balancing decisions on the packet rate metrics, but the Host screen displays all of the metrics for your convenience.

Configuration issues

The SNMP probing feature requires that each host run an SNMP agent, and that there is open network communication between the hosts and the **big3d** agents in the data centers. Certain firewall configurations block SNMP communications, and you may need to verify that the firewalls in your network allow SNMP traffic to pass through. The 3DNS Controller supports the following common SNMP agents for host probing:

- ❖ **Generic**

The 3DNS Controller can work with a generic SNMP agent running on a host.

- ❖ **UCD SNMPD**

The UCD SNMPD is a free SNMP agent provided by the University of California at Davis. It is freely available on the web at <http://ucd-snmp.ucdavis.edu>, <ftp://ucd-snmp.ucdomain.edu/ucd-snmp.tar.gz>.

- ❖ **Solstice Enterprise**

The Solstice Enterprise agent is a product of SunSoft.

- ❖ **Windows NT 4.0 SNMP**

The Windows NT 4.0 SNMP matrix agent is distributed with the Microsoft Windows NT 4.0 server.

In addition to properly configuring these agents on the hosts themselves, you need to specify SNMP host probing settings in two places in the 3DNS Controller configuration. First, when you define a BIG/ip Controller or 3DNS Controller server, you set the **big3d** agent to run at least one SNMP factory. Second, when you define the host servers, you configure specific SNMP agent settings for each host. For example, you need to specify the type of agent running on the host as well as the community string that allows access to the SNMP agent.

Planning sync groups

A **sync group** is a group of 3DNS Controllers that share information. In a sync group, a **principal** 3DNS Controller issues requests to the **big3d** agents to gather metrics data. Both the principal controller and the **receiver** 3DNS Controllers in the group

receive broadcasts of metrics data from the **big3d** agents. All controllers in the group also receive broadcasts of updated configuration settings from the 3DNS Controller that has the latest configuration changes.

When you define the sync group, select 3DNS Controllers from the list of servers you have already defined. The sync group lists the controllers in the order in which you selected them. The first controller in the list is the principal 3DNS Controller. The remaining controllers in the list are receiver 3DNS Controllers. If the principal controller goes **down**, the next controller in the list becomes the principal 3DNS Controller until the original principal controller comes back online.

Understanding how sync groups work

The sync group feature synchronizes individual configuration files, such as **wideip.conf** and other files that store system settings. You have the option of adding files to the synchronization list.

The controllers in a sync group actually operate as peer servers. At set intervals, the **sync** daemon compares the timestamps of the config files earmarked for synchronization on all of the controllers. If the timestamps on a specific file differ between controllers, the controller with the latest file broadcasts the file to all of the other controllers in the group.

Setting up communications between 3DNS Controllers, BIG/ip Controllers, and big3d agents

There are three different communication issues that you need to resolve when you set up communication between controllers running in your network:

❖ 3DNS Controllers communicating with other 3DNS Controllers

To allow 3DNS Controllers to communicate with each other, you need to set up **ssh** and **scp** tools for controllers that communicate

with other controllers within the US, and **rsh** and **rcp** tools for controllers that communicate with other controllers outside the US.

❖ **3DNS Controllers communicating with BIG/ip Controllers**

To allow 3DNS Controllers to communicate with BIG/ip Controllers, you address the same **ssh** and **rsh** issues. Controllers communicating within the US can use **ssh** and **scp**, but controllers communicating outside the US require **rsh** and **rcp**.

❖ **3DNS Controllers communicating with big3d agents**

To allow communications between **big3d** agents and the 3DNS Controller, you need to configure iQuery ports on both the 3DNS Controllers and the BIG/ip Controllers that run the **big3d** agent.

◆ **Note**

*Enabling **rsh** and **rcp** does not prevent US 3DNS Controllers from using encryption when they communicate with other US 3DNS Controllers.*

Figure 2.1 illustrates the communication links between 3DNS Controllers and other machines in your network. The following sections describe in detail which ports and protocols are used for specific communication links.

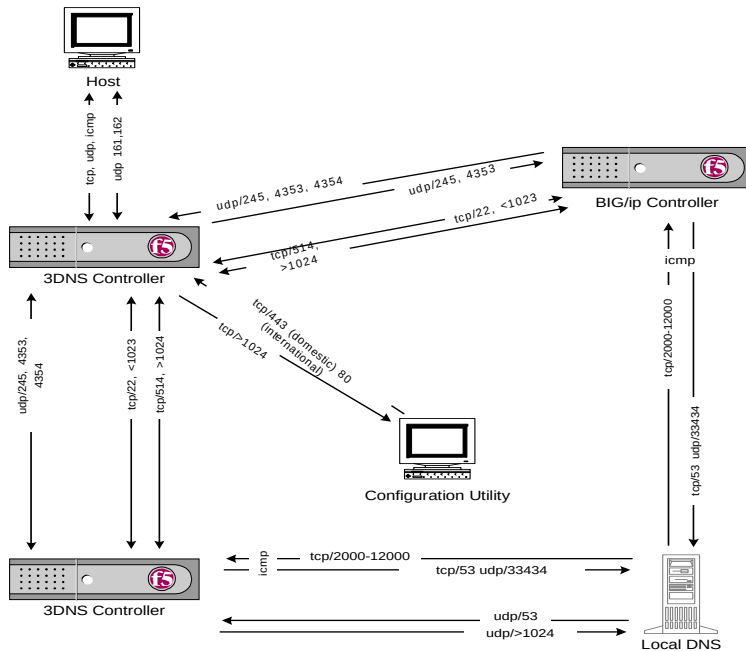


Figure 2.1 Communication ports

Setting up communication between 3DNS Controllers

3DNS Controllers need to communicate with each other in order to synchronize configuration and performance data. If you use 3DNS Controllers exclusively inside the US, or exclusively outside the US, the communication tools set up by the First-Time Boot utility are all you need. US controllers all use **ssh** and **scp**, and international controllers all use **rsh** and **rcp**.

If you work in a mixed environment where some 3DNS Controllers are in the US and other 3DNS Controllers are outside the US, you need to enable the **rsh** and **rcp** tools on the US 3DNS Controllers. These tools come pre-installed on the US 3DNS Controllers, but you must explicitly enable the **rsh** and **rcp** tools on the US 3DNS

Controllers. You can easily enable these tools by running the **rsetup** script from the command line, by running **config-rsd** script, or when you run the First-Time Boot utility.

Table 2.2 shows the ports and protocols that 3DNS Controllers use to communicate with each other.

From	To	Protocol	From Port	To Port	Purpose
US 3DNS Controller	US 3DNS Controller	tcp	<1023	22	SSH/SCP
International 3DNS Controller	International 3DNS Controller	tcp	>1024	514	RSH/RCP
US 3DNS Controller	International 3DNS Controller	tcp	>1024	514	RSH/RCP
International 3DNS Controller	US 3DNS Controller	tcp	>1024	514	RSH/RCP

Table 2.2 Communications between 3DNS Controllers

Setting up communication between 3DNS Controllers and BIG/ip Controllers

3DNS Controllers need to communicate with BIG/ip Controllers in order to copy **big3d** agents from the 3DNS Controllers to the BIG/ip Controllers. If you use 3DNS Controllers and BIG/ip Controllers exclusively inside the US, or exclusively outside the US, the communication tools set up by the First-Time Boot utility are all you need. US controllers always use **ssh** and **scp**, and international controllers always use **rsh** and **rcp**.

However, if you work in a mixed environment where some controllers are in the US and other controllers are outside the US, you need to enable the **rsh** and **rcp** tools on the US controllers. These tools come pre-installed on both US 3DNS Controllers and US BIG/ip Controllers, but you must explicitly enable them as described below.

To enable the rlogin tools on a BIG/ip Controller

From the command line, run the **rsetup** script.

◆ Note

*You can disable **rsh** and **rcp** access at any time by changing the **bigip.open_rsh_ports** system control variable to **0** in **/etc/rc.sysctl**.*

Table 2.3 shows the ports and protocols that the 3DNS Controller uses to communicate with BIG/ip Controllers. For more information about using **rsh** and **rcp** tools on the BIG/ip Controller, refer to the **BIG/ip Controller Administrator Guide**.

From	To	Protocol	From Port	To Port	Purpose
US 3DNS Controller	US BIG/ip Controller	tcp	<1023	22	SSH/SCP
International 3DNS Controller	International BIG/ip Controller	tcp	>1024	514	RSH/RCP
US 3DNS Controller	International BIG/ip Controller	tcp	>1024	514	RSH/RCP
International BIG/ip Controller	US 3DNS Controller	N/A	N/A	N/A	N/A

Table 2.3 Communications between 3DNS Controllers and BIG/ip Controllers

Setting up iQuery communications for the big3d agent

The iQuery protocol can use one of two ports to communicate between the **big3d** agents and the 3DNS Controllers. The ports used by iQuery traffic change, depending on whether the traffic is inbound from the **big3d** agent or outbound from the 3DNS Controller.

Table 2.4 shows the port numbers and corresponding protocols used for iQuery traffic.

From	To	Protocol	From Port	To Port	Purpose
3DNS Controller	big3d agent	udp	245, 4354	245	Standard iQuery port for outbound traffic
3DNS Controller	big3d agent	udp	4353 or 4354	4353	Alternate iQuery port for outbound traffic (open this port only when the use_alternate_iq global variable is set to yes)
big3d agent	3DNS Controller	udp	245 or 4353	4354	Ephemeral port used for inbound iQuery traffic
big3d agent	3DNS Controller	udp	245	245	Single port used for multiplexed inbound iQuery traffic (open this port only when the multiplex_iq global variable is set to yes)
big3d agent	3DNS Controller	udp	4353	4353	Single port used for multiplexed inbound iQuery traffic (open this port only when both the use_alternate_iq and the multiplex_iq global variables are set to yes)
big3d agent	host SNMP agent	udp	>1024	161	Ephemeral ports used to make SNMP queries for host statistics
host SNMP agent	big3d agent	udp	161	>1024	Ephemeral ports used to receive host statistics via SNMP

Table 2.4 Communications between 3DNS Controllers, **big3d** agents, and host servers

Note that if you run **big3d** agents in a mixed US/international environment, the US controllers automatically turn off Blowfish encryption when communicating with international **big3d** agents.

When communicating with US **big3d** agents, however, US 3DNS Controllers always use Blowfish encryption by default, though you can manually disable it if you prefer.

Allowing iQuery communications to pass through firewalls

The payload information of an iQuery packet contains information that potentially requires translation when there is a firewall in the path between the **big3d** agent and the 3DNS Controller. Only packet headers are translated by the firewall, payloads are not.

The iQuery translation option resolves this issue. With iQuery translation turned on, the iQuery packet stores the original IP address in the packet payload itself. When the packet passes through a firewall, the firewall translates the IP address in the packet header normally, but the IP address within the packet payload is preserved. The 3DNS Controller reads the IP address out of the packet payload, rather than out of the packet header.

In the example configuration shown in Figure 2.2, a firewall separates the path between a BIG/ip Controller running a **big3d** agent and the 3DNS Controller. The packet addresses are translated at the firewall. However, addresses within the iQuery payload are not translated and they arrive at the BIG/ip Controller in their original states.

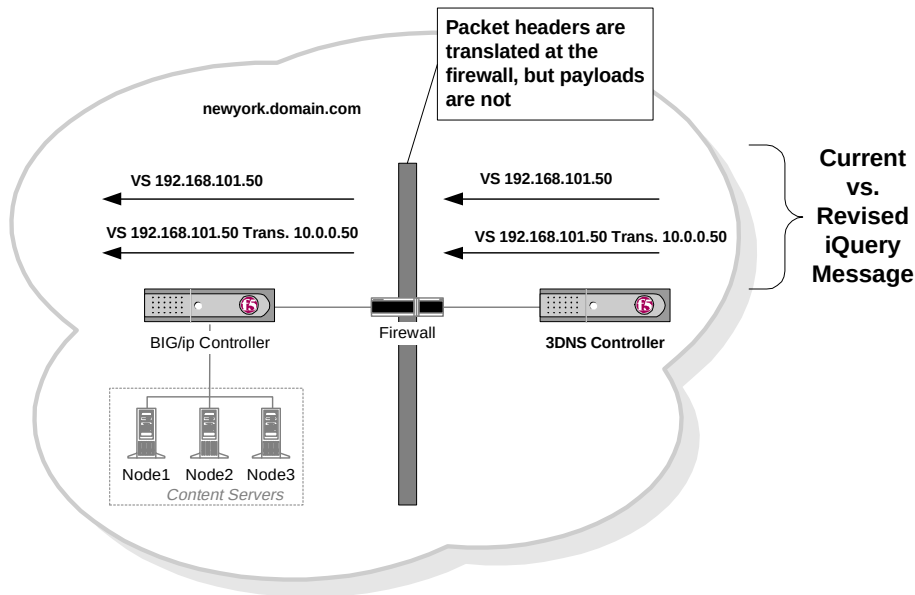


Figure 2.2 Translating packet address the firewall

Communications between 3DNS Controllers and other machines in the network

The following tables show the other ports and protocols that 3DNS Controller uses for communication. Table 2.5 shows the ports that the 3DNS Controller uses for remote administrative connections to the 3DNS web server.

From	To	Protocol	Port	Purpose
Remote Workstation	US 3DNS Controller	tcp	443	Connection to secure web server
Remote Workstation	Int'l 3DNS Controller	tcp	80	Connection to standard web server

Table 2.5 Communications between 3DNS Controllers and remote workstations

Table 2.6 shows the ports on which the 3DNS Controller receives and responds to DNS resolution requests issued by local DNS servers.

From	To	Protocol	From Port	To Port	Purpose
LDNS	3DNS Controller	udp	53 or >1024	53	DNS resolution requests
3DNS Controller	LDNS	udp	53	53 or >1024	DNS resolution answers

Table 2.6 *DNS communications on the 3DNS Controller*

Table 2.7 shows the ports that the **big3d** agent uses when collecting path data for LDNS servers.

From	To	Protocol	From Port	To Port	Purpose
big3d agent	LDNS	icmp	N/A	N/A	Probing using ICMP pings
big3d agent	LDNS	tcp	2000-12000	53	Probing using TCP (CISCO routers should "allow establish")
LDNS	big3d agent	tcp	53	2000-12000	Probing using TCP (CISCO routers should "allow establish")
big3d agent	LDNS	udp	2000-12000	33434	UDP probing and traceroute
LDNS	big3d agent	icmp	N/A	N/A	Replies from ICMP, UDP pings, or traceroute probes

Table 2.7 *Communication between **big3d** agents and LDNS servers*

The **big3d** agent can run on either a 3DNS Controller or a BIG/ip Controller. If you run a **big3d** agent on a BIG/ip Controller and you set the SNMP probing factory count to **1** or higher, the **big3d**

agent automatically opens UDP ports to allow for SNMP communications. If you do not want to open UDP ports for this purpose, you need to set the SNMP factory count to **0**.

Planning issues for the load balancing configuration

The final phase of installing 3DNS Controllers is setting up the load balancing configuration. Load balancing configurations are based on pools of virtual servers. When the 3DNS Controller receives a connection request, it uses a load balancing mode to determine which virtual server in a given pool should receive the connection. The virtual servers in the pool can be the virtual servers managed by BIG/ip Controllers, or virtual servers managed by a generic host server, or they can be individual host servers themselves. Note that the 3DNS Controller continuously verifies which virtual servers in the pool are currently available to accept load balanced connections.

Simple configurations typically use a single pool of virtual servers and a load balancing mode that does not require significant additional configuration steps, such as Round Robin or Hops. More advanced load balancing configurations can use multiple pools, customizable load balancing modes, and other advanced traffic control features, such as topology-based access control and production rules. If you plan on implementing a more complex configuration, you may want to refer to Chapter 6 for additional details about advanced load balancing features.

Understanding the wideip key

The ***wideip key*** is the same address as the domain name. The wide IP key binds the information from DNS to the 3DNS Controller and indicates to DNS that the 3DNS Controller (within the **named** process) should attempt to handle requests to this domain name.

This allows the 3DNS Controller to resolve the request by making a decision based upon its metric database and returning a better answer. Each wide IP definition must have its own unique address.

The wide IP key is sometimes referred to as the fallback address. When the preferred, alternate, and fallback load balancing modes (as specified in the wideip definition) fail, the 3DNS Controller instructs the DNS to issue its original answer. When this happens, the wide IP key is called the fallback address.

Choosing a load balancing mode

The 3DNS Controller performs load balancing by selecting one server from a pool of servers available for load balancing. When the 3DNS Controller receives a name resolution request, the controller uses a load balancing mode to select the best available virtual server from the pool. Once the 3DNS Controller selects the virtual server, it constructs the DNS answer, an **A** record, and sends the answer back to the requesting client's local DNS server.

The 3DNS Controller can choose virtual servers from the pool using either a static load balancing mode, which selects a server based on a pre-defined pattern, or a dynamic load balancing mode, which selects a server based on current performance.

The 3DNS Controller uses load balancing modes in two places:

❖ **Load balancing within a pool**

Within each pool, the 3DNS Controller allows you to specify three different load balancing modes that the controller uses in sequential order. The 3DNS Controller first uses the load balancing mode specified as the ***preferred*** method. If this load balancing mode fails, the 3DNS Controller then uses the load balancing mode specified as the ***alternate*** method. If that load balancing mode fails as well, the 3DNS Controller uses the load balancing mode specified as the ***fallback*** method. Only if the fallback method fails does the 3DNS Controller return the client to standard DNS for resolution.

❖ **Load balancing among multiple pools**

The 3DNS Controller supports multiple pools. Configurations that contain two or more pools use a load balancing mode

between the different pools. Once a pool has been selected, the 3DNS Controller uses a load balancing mode to choose a virtual server within the selected pool.

Table 2.8 shows a complete list of supported load balancing modes, and where you can use each mode in the 3DNS Controller configuration. Note that the following sections describe how each of these load balancing modes works.

Load Balancing mode	Pool load balancing	Preferred	Alternate	Fallback
Completion Rate		x		x
Global Availability	x	x	x	x
Hops		x		x
Least Connections		x	x	x
Null			x	x
Packet Rate		x	x	x
Quality of Service		x		x
Random	x	x	x	x
Ratio	x	x	x	x
Return to DNS		x*	x	x
Round Robin	x	x	x	x
Round trip time		x		x
Topology		x	x	x

Table 2.8 Load balancing mode usage

Using static modes

Static load balancing modes distribute connections across the network according to predefined patterns, and take server availability into account. The 3DNS Controller supports the following static load balancing modes:

- ❖ Round Robin
- ❖ Ratio
- ❖ Random
- ❖ Global Availability
- ❖ Topology
- ❖ Null
- ❖ Return to DNS

The Null and Return to DNS load balancing modes are special modes that you can use to skip load balancing under certain conditions. The remaining static load balancing modes perform true load balancing as described in the following sections.

Round Robin mode

Round Robin mode distributes connections in a circular and sequential pattern among the virtual servers in a pool. Over time, each virtual server receives an equal number of connections.

Figure 2.3 shows a sample of the connection distribution pattern for Round Robin mode.

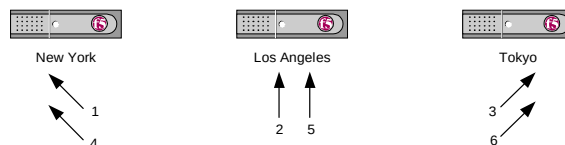


Figure 2.3 Round Robin mode

Ratio mode

Ratio mode distributes connections among a pool of virtual servers as a weighted Round Robin. For example, you can set up Ratio mode to send twice as many connections to a fast, new server, and only half as many connections to an older, slower server.

This load balancing mode requires that you define a ratio weight for each virtual server in a pool, or for each pool if you are using Ratio mode to do load balancing among multiple pools. The default ratio weight for a server or a pool is set to **1**.

Figure 2.4 shows a sample connection distribution for Ratio mode.

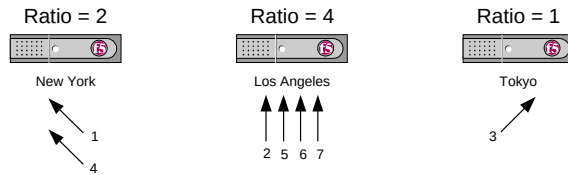


Figure 2.4 Ratio mode

Random mode

Random mode sends connections to virtual servers in a random pattern.

Global Availability mode

Global Availability mode uses the virtual servers included in the pool in the order in which they are listed. For each connection request, this mode starts at the top of the list and sends the connection to the first available virtual server in the list. Global Availability mode moves to the next virtual server in the list only when the current virtual server is full or otherwise unavailable. Over time, the first virtual server in the list receives the most connections and the last virtual server in the list receives the least number of connections.

Topology mode

Topology allows you to direct or restrict traffic flow by entering network information into the configuration file. This allows you to develop proximity-based mapping. For example, customers in a particular geographic region can be sent to servers within that same region. The 3DNS Controller determines the proximity of servers by comparing the client's LDNS IP address to the IP address of the available servers.

This load balancing mode requires you to do some advanced configuration planning, such as gathering the information you need to define the topology records which determine proximity of client LDNS servers to the various virtual servers.

The Topology load balancing mode is different from the topology-based access control feature. Topology-based access control actually prevents clients from connecting to specific virtual servers. You can use the topology-based access control feature in conjunction with the Topology load balancing mode. See Chapter 6, *Configuring Specialized Load Balancing*, for detailed information about working with this and other topology features.

Null mode

The Null load balancing mode is a special mode that you can use if you want to skip the current load balancing method, or skip to the next pool in a multiple pool configuration. For example, if you set an alternate method to Null in a pool, the 3DNS Controller skips the alternate method and immediately tries the load balancing mode specified as the fallback method. If the fallback method is set to Null, the 3DNS Controller either uses the next pool, if you have multiple pools, or it returns the connection request to DNS for resolution.

This mode is most useful for multiple pool configurations. For example, you can temporarily remove a specific pool from service by setting each of the methods (preferred, alternate, and fallback) to Null. You could also use the mode to limit each pool to a single load balancing mode. For example, you would set the preferred method in each pool to the desired load balancing mode, and then you would set both the alternate and fallback methods to Null in

each pool. If the preferred method failed, the Null mode in both the alternate and fallback methods would force the 3DNS Controller to go to the next pool for a load balancing answer.

Return to DNS mode

The Return to DNS mode is another special load balancing mode that you can use to immediately return connection requests to DNS for resolution. This mode is particularly useful if you want to temporarily remove a pool from service, or if you want to limit a pool in a single pool configuration to only one or two load balancing attempts.

Using dynamic load balancing modes

Dynamic load balancing modes distribute connections to servers that show the best current performance. The performance taken into account depends on the particular dynamic mode you are using.

All dynamic load balancing modes make load balancing decisions based on the metrics collected by the **big3d** agents running in each data center. The **big3d** agents collect the information at set intervals that you can define when you set the global **ttd** variables.

The 3DNS Controller supports the following dynamic load balancing modes:

- ❖ Completion Rate
- ❖ Least Connections
- ❖ Packet Rate
- ❖ Round Trip Times (RTT)
- ❖ Hops
- ❖ Quality of Service

Completion Rate mode

Completion Rate mode selects a virtual server that currently maintains the least number of dropped or timed out packets for transactions between a data center and the client LDNS.

Figure 2.5 shows a sample connection distribution pattern for Completion Rate mode.

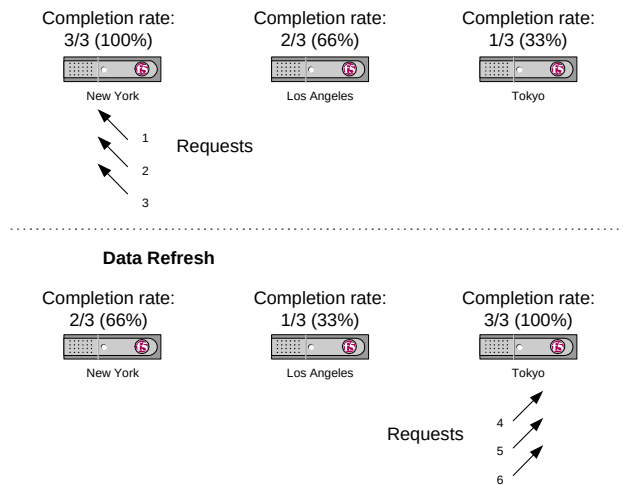


Figure 2.5 Completion Rate mode

Least Connections mode

Least Connections mode is also used for load balancing virtual servers managed by BIG/ip Controllers. Least Connections mode simply selects a virtual server on the BIG/ip Controller that currently hosts the fewest connections.

Packet Rate mode

Packet Rate mode selects a virtual server that is currently processing the fewest number of packets per second.

Figure 2.6 shows a sample connection distribution for Packet Rate mode.

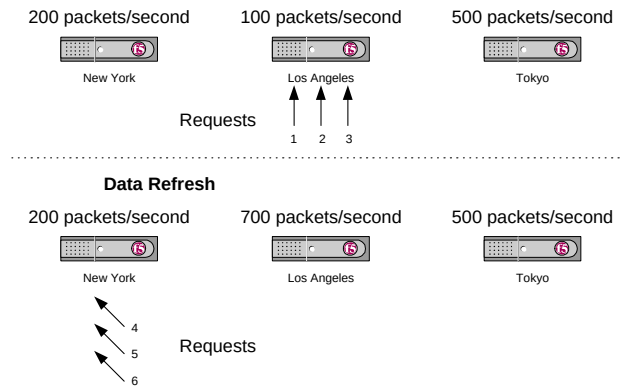


Figure 2.6 Packet Rate mode

Round Trip Times mode

Round Trip Times (RTT) mode selects the virtual server with the fastest measured round trip time between the data center and the client LDNS. This load balancing mode requires that you run one or more **big3d** agents in each data center to collect the required metrics.

Figure 2.7 shows a sample connection distribution for Round Trip Times mode.

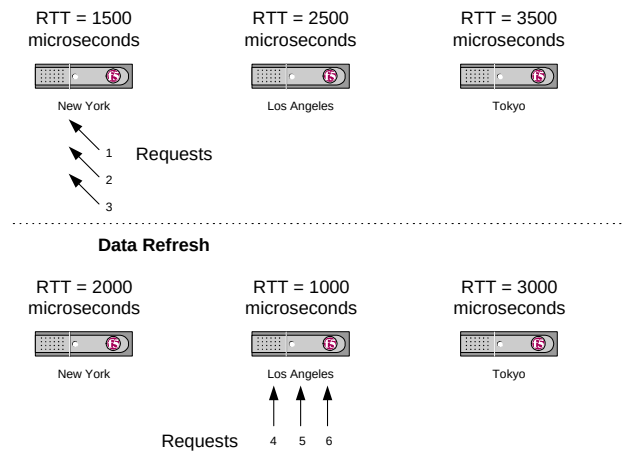


Figure 2.7 Round Trip Times mode

Hops mode

Hops mode is based on the traceroute utility, and it tracks the number of intermediate system transitions between the client LDNS and each data center. Hops mode selects a virtual server in the data center that has the fewest network hops.

Quality of Service mode

Quality of Service mode uses the current performance information, calculates an overall score for each virtual server, and then distributes connections based on each virtual server's score. The performance factors that it takes into account include:

- ❖ Round trip time
- ❖ Hops
- ❖ Completion rate
- ❖ Packet rate
- ❖ Topology

Quality of Service mode is a customizable load balancing mode. For simple configurations, you can easily use the mode with its default settings where all the factors are weighted equally. For more advanced configurations, you can specify different weights for each performance factor in the equation.

You can also configure the Quality of Service load balancing mode to use the dynamic ratio feature. With the dynamic ratio feature turned on, the Quality of Service mode becomes similar to the Ratio mode where the connections are distributed in proportion to ratio weights assigned to each virtual server. The ratio weights are based on the **qos** scores; the better the score, the higher percentage of connections the virtual server receives.

For details about customizing Quality of Service mode, see Chapter 6, *Configuring Specialized Load Balancing*.

Ensuring availability for e-commerce, FTP, and other services that use multiple ports

Before the 3DNS Controller selects a server to receive a connection, it verifies that the server is up and available. Certain types of network traffic, such as FTP traffic or e-commerce traffic, require that more than one port be available in order for the client's requests to be properly handled. For example, FTP servers use both ports 20 and 21, while e-commerce sites typically require that both ports 80 and 443 are available to handle HTTP and SSL traffic.

When you set up a load balancing configuration, you can define a list of ports that are verified on each virtual server before the server is made available to receive load balanced connections.

Using the LDNS round robin wide IP attribute

LDNS Round Robin is an attribute that you can use in conjunction with any load balancing mode. The LDNS round robin attribute allows the 3DNS Controller to return a list of available virtual servers, instead of a single virtual server. Certain browsers keep the answer returned by DNS servers. By enabling this attribute, the 3DNS Controller returns a maximum of 16 virtual servers as the answer to a DNS resolution request. This provides browsers with alternate answers when a virtual server becomes unavailable.

Using advanced traffic control features

The 3DNS Controller offers three advanced features that you can configure to further control the distribution and flow of network traffic:

- ❖ **Topology-based access control**

Topology-based access control allows you to restrict clients from connecting to virtual servers in specific data centers.

- ❖ **IP packet filtering**

You can use IP packet filtering on the 3DNS Controller to reject connection requests from certain source IP addresses.

- ❖ **Production rules**

The production rules feature allows you to change the load balancing configuration, as well as other system settings, based on dynamic factors such as current network performance, as well as time of day.

Configuring topology-based access control

Topology-based access control limits users to a subset of available servers based on their proximity to the servers. You can use topology-based access control to force European clients to connect only to servers also in Europe, rather than connecting to servers in South America.

The 3DNS Controller uses topology-based access control in conjunction with load balancing modes. For example, even though topology-based access control may force a European client to connect to a server in Europe, the 3DNS Controller still uses the load balancing modes specified for the preferred, alternate, and fallback methods to choose the best European server that should receive the connection.

For details about working with topology-based features and creating the necessary topology records, see *Chapter 6, Configuring Specialized Load Balancing*.

Setting up IP packet filtering

You can use the IP packet filtering feature to reject unwanted connection requests that may bog down your site and compromise performance. For example, if you detect an attack on your site, such as when your system is suddenly flooded with DNS requests from a single client, you can block that client's connection requests by defining an IP filter that rejects all packets containing that client's source IP address.

Note that the IP packet filtering supported on the 3DNS Controller is based on BSD IP packet filtering. You can configure IP filters manually, but we recommend that you define IP filters in the F5 Configuration utility, which greatly simplifies the process. To define a new IP filter in the Configuration utility, you specify one of three filter criteria:

- ❖ Source IP address only
- ❖ Destination IP address only
- ❖ Combination of source and destination IP address

Note that you can use IP address ranges when you define IP filters. You can find details about how to configuring IP filters in Chapter 7, *Monitoring and Administration*.

Defining production rules

Production rules are a policy-based management feature that you can use to dynamically change the load balancing configuration and the system settings based on specific triggers, such as the time of day, or the current network traffic flow. You can set up standard production rules using the F5 Configuration utility, or you can define custom production rules using the production rules scripting language.

Refer to Chapter 7, *Monitoring and Administration*, for information about setting up production rules.

Planning DNS zone file management

An important part of installing 3DNS Controllers in your network is planning which servers should be master for a given DNS zone. When you initially set up a 3DNS Controller in your network, you have two basic options for setting up DNS zone masters:

- ❖ You can use the 3DNS Controller as the master DNS server for your domain.
- ❖ You can use an existing master DNS for your domain, and use the 3DNS Controller only as the master DNS for your wide IP sub-domains.
- ❖ The 3DNS Controller must always be the DNS master for your wide IP sub-domains, regardless of which server is the master DNS in your network. We strongly recommend that you set up the 3DNS Controller to run as a master DNS and manage your domain.

One major benefit of setting up the 3DNS Controller to be the master DNS for your domain is that you can easily manage DNS zone files using NameSurfer, a browser-based third-party

application included on the 3DNS Controller. You can also easily transfer your existing zone files to the 3DNS Controller after the initial installation.

When you define wide IPs in the F5 Configuration utility, the NameSurfer application automatically makes the appropriate additions to the zone files and broadcasts the new zone files to the other DNS servers in your network. If you configure wide IPs manually, however, you need to make the corresponding zone files changes manually.

If you choose to use the advanced synchronization features of 3DNS, we strongly recommend that you configure each 3DNS Controller to run as a master DNS. This effectively creates a group of peer master DNS servers. This type of configuration offers the following advantages:

- ❖ You can change zone files on any one of the 3DNS Controllers in the network and have those changes automatically broadcast to all of the other controllers in the network.
- ❖ Each 3DNS Controller has the most up-to-date zone files, providing you one or more layers of redundancy.
- ❖ The NameSurfer application automatically controls the addition, configuration, and deletion of zone files.

Replacing your DNS servers with 3DNS Controllers as master DNS servers for your domain

Figure 2.8 shows an implementation where both 3DNS Controllers in the network act as master DNS servers for the domain, **domain.com**.

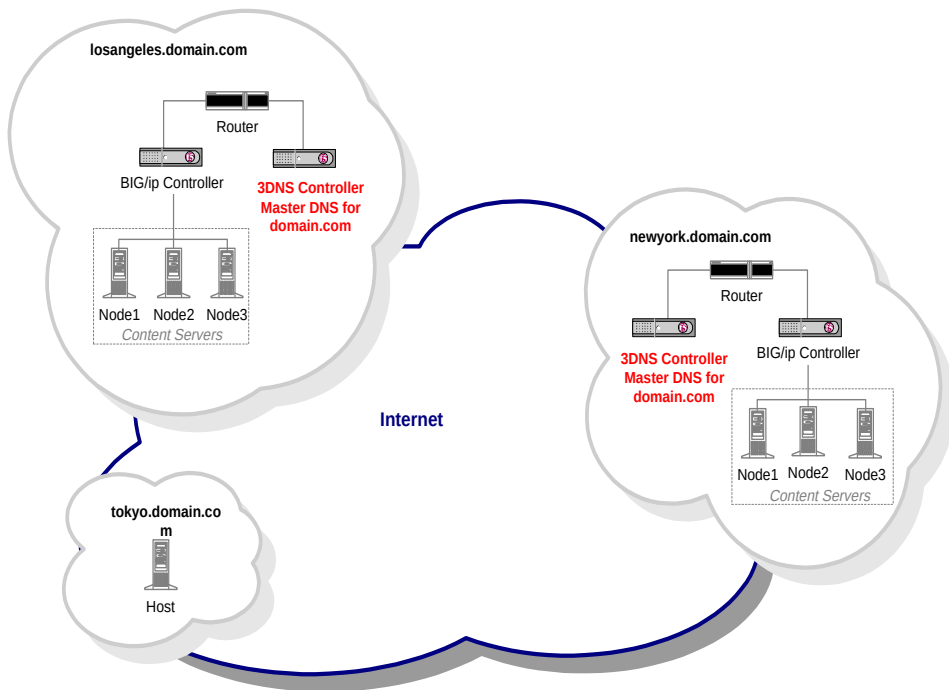


Figure 2.8 3DNS Controllers as DNS masters for your domain

Converting existing BIND files during an initial installation

After you initially install the 3DNS Controller, you need to transfer existing BIND files, and then convert them to the NameSurfer format.

One option for converting your existing BIND files is to skip the NameSurfer configuration when you run the First-Time Boot utility. You transfer the zone files and **named.conf** file after the system has rebooted and then run the **config_namesurfer** script that configures, converts, and starts the NameSurfer application. For more information, see *Appendix C, BIND 8 Information*.

The second option for importing your existing BIND files is to zone transfer from your current name server to NameSurfer. After configuring NameSurfer during the First-Time Boot utility, and connecting to the Configuration utility, use the **Copy from other name server** option in the NameSurfer UI. For more information, refer to the NameSurfer documentation.

Running 3DNS Controllers as DNS masters only for wide IP sub-domains

At a minimum, all 3DNS Controllers must be the DNS masters for the zone files associated with wide IP definitions. When you set up a configuration where the 3DNS Controllers are DNS master only for those sub-domains, you need to make a few changes to the zone files on the master DNS for your domain after the configuring your 3DNS Controller.

Figure 2.9 shows an example where both 3DNS Controllers are DNS masters for the wide IP sub-domains, and a generic name server in the Tokyo data center is the master DNS server for the domain, **domain.com**.

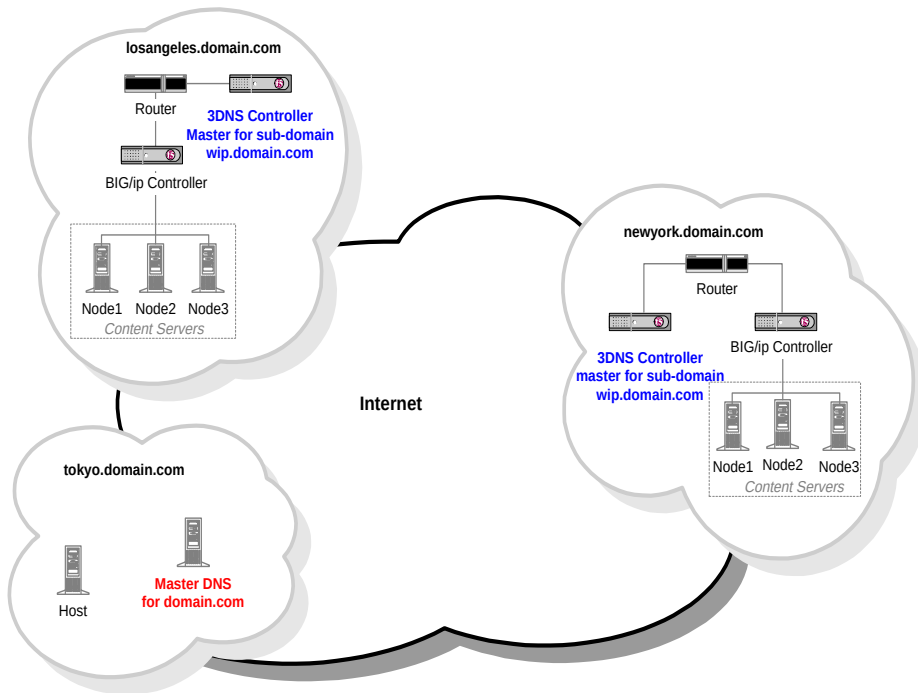


Figure 2.9 3DNS Controllers managing sub-domains

Note that you still set NameSurfer to be the master during the First-Time Boot utility for initial installations, or during the NameSurfer configuration script for upgrade installations. Remember that NameSurfer is the master for the zone files on the 3DNS Controller, but in this configuration the zone files contain only those records associated with wide IPs configured on the 3DNS Controller. When you configure wide IPs in the F5 Configuration utility, the NameSurfer application automatically updates the corresponding sub-domain zones and broadcasts them to the other DNS servers in the network. For configurations where synchronization is enabled, all changes to any NameSurfer files are automatically updated to the other 3DNS Controllers.



3

Setting up the Hardware

- **Unpacking and installing the hardware**
- **Running the First-Time Boot utility**
- **Enabling remote login tools**
- **Preparing workstations for command line access**



Unpacking and installing the hardware

There are two basic tasks required to get the 3DNS Controller installed and set up. First, you need to connect the peripheral hardware and connect the 3DNS Controller to the network, and then you need to turn the system on and run the First-Time Boot utility. The First-Time Boot utility is a wizard that helps you configure basic system elements such as administrative passwords, IP addresses, and host names for both the root system and for the 3DNS web server. Once you complete the First-Time Boot utility, you can continue the configuration process either from a remote administrative workstation, or from the console itself.

Reviewing the hardware requirements

The 3DNS Controller comes with the separate hardware pieces that you need for installation and maintenance. You must provide standard peripheral hardware, such as a keyboard or serial terminal.

Hardware provided with the 3DNS Controller

When you unpack the 3DNS Controller, you should make sure that the following components are included:

- ❖ One power cable
- ❖ One PC/AT-to-PS/2 keyboard adapter
- ❖ Four rack-mounting screws
- ❖ Two keys for the front panel lock
- ❖ One extra fan filter
- ❖ One ***3DNS Controller Administrator Guide***

If you purchased a hardware-based redundant system, you also receive one fail-over cable to connect the two controller units together (network-based redundant systems do not require a fail-over cable). Additionally, if you purchased a US 3DNS Controller that supports encryption, you receive the *F-Secure SSH Client* manual, published by Data Fellows.

Peripheral hardware that you provide

For each 3DNS Controller in the system, you need to provide the following peripheral hardware:

- ❖ You need standard input/output hardware for direct administrative access to the 3DNS Controller. Either of the following options is acceptable:
  - A VGA monitor and PC/AT-compatible keyboard
  - A serial terminal and a null modem cable (see *To configure a serial terminal in addition to the console*, on page 3-7, for serial terminal configuration information)
- ❖ You also need network hubs, switches, or concentrators to connect to the 3DNS Controller network interfaces. The devices you select must be compatible with the network interface cards installed in your 3DNS Controller. The devices can support 10/100 Ethernet, Gigabit Ethernet and FDDI/CDDI (including multiple FDDI and full duplex).
  - For Ethernet, you need either a 10Mb/sec or 100 Mb/sec hub or switch.
  - For FDDI/CDDI, a concentrator or a switch is optional.

If you plan on doing remote administration from your own PC workstation as most users do, we recommend that you have your workstation already in place. Keep in mind that the First-Time Boot utility prompts you to enter your workstation's IP address when you set up remote administrative access.

Familiarizing yourself with the 3DNS Controller hardware

Before you begin to install the 3DNS Controller, you may want to quickly review the following figures that illustrate the controls and ports on both the front and the back of a standard 3DNS Controller. If you have a special hardware configuration, the ports on the back of your unit will differ slightly from those shown below.

◆ **Note**

The ports on the back of every 3DNS Controller are individually labeled, so it should be clear what each port is, no matter which hardware configuration you have purchased.

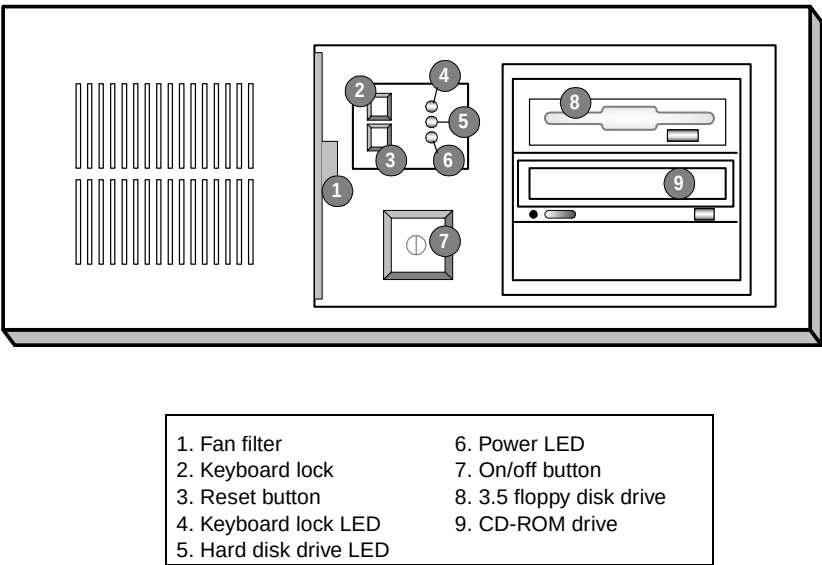
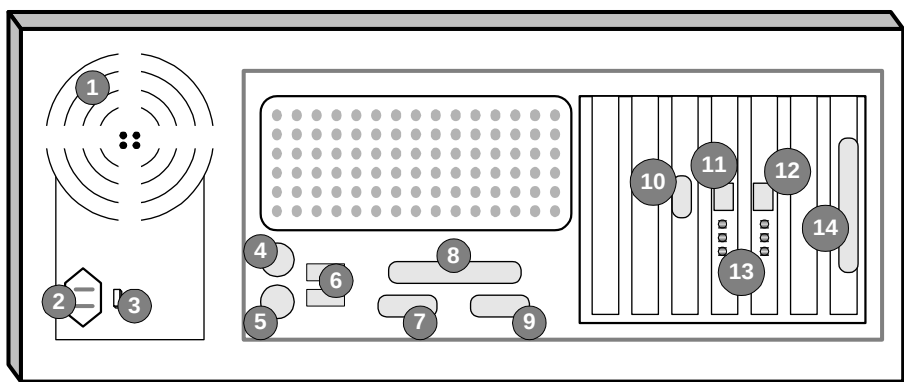


Figure 3.1 Front view of a 3DNS Controller

Figure 3.1 illustrates the front of a 3DNS Controller with the access panel open. On the front of the unit, you can turn the unit off and on, or you can reset the unit. You can also view the indicator lights for hard disk access and for the keyboard lock.

Figure 3.2, the following figure, illustrates the back of a 3DNS Controller. Note that all ports are labeled, even those which are not intended to be used with the 3DNS Controller. Ports marked with an asterisk (*) in the list following are not used by the 3DNS Controller, and do not need to be connected to any peripheral hardware.



1. Fan	8. Printer port*
2. Power in	9. Fail-over port
3. Voltage selector	10. Video (VGA) port
4. Mouse port*	11. Secondary Ethernet interface (RJ-45)
5. Keyboard port	12. Primary Ethernet interface (RJ-45)
6. Universal serial bus ports*	13. Interface indicator LEDs
7. Serial terminal port	14. Watchdog card*

*Not to be connected to any peripheral hardware.

Figure 3.2 Rear view of a 3DNS Controller

Environmental requirements and usage guidelines

A 3DNS Controller is an industrial network appliance, designed to be mounted in a standard 19-inch rack. To ensure safe installation and operation of the unit, be sure to consider the following before you install the unit in the rack:

- ❖ Always install the rack according to the manufacturer's instructions, and be sure to check the rack for stability before placing equipment in it.
- ❖ Build and position the rack so that once you install the 3DNS Controller, the power supply and the vents on both the front and back of the unit remain unobstructed. The 3DNS Controller must have adequate ventilation around the unit at all times.
- ❖ Do not allow the air temperature in the room to exceed 50° C. The internal temperature of the room where the 3DNS Controller is located should be considered for continued safe operation.
- ❖ Make sure that the branch circuit into which you plug the unit is not shared by more electronic equipment than it is designed to manage safely at one time.
- ❖ If you are installing the 3DNS Controller in a location outside of the United States, you need to verify that the voltage selector is set appropriately before connecting the power cable to the unit.

◆ WARNING

The unit must be connected to Earth ground, and it should have a reliable ground path maintained at all times.

◆ WARNING

The 3DNS Controller contains a lithium battery. There is danger of an explosion if you replace the lithium battery incorrectly. We recommend that you replace the battery only with the same type of battery originally installed in the unit, or with an equivalent type recommended by the battery manufacturer. Be sure to discard all used batteries according to the manufacturer's instructions.

Installing and connecting the hardware

There are six basic steps to installing the hardware. You simply need to install the controller in the rack, connect the peripheral hardware and the network interfaces, and then connect the fail-over and power cables.

WARNING

You should not turn on a 3DNS Controller until all peripheral hardware is connected to the unit.

To install the hardware

1. Insert the 3DNS Controller in the rack and secure it using the four rack-mounting screws that are provided.
2. Connect the hardware that you have chosen to use for input/output:
  - If you are using a VGA monitor and keyboard, connect the monitor connector cable to the video port (number 10 in Figure 3.2, on page 3-4) and the keyboard connector cable to the keyboard port (number 5 in Figure 3.2, on page 3-4). Note that a PC/AT-to-PS/2 keyboard adapter is included with each 3DNS Controller (see the component list on page 3-2).
  - Optionally, if you are using a serial terminal as the console, connect the serial cable to the terminal serial port (number 7 in Figure 3.2). Also, you should not connect a keyboard to the 3DNS Controller. When there is no keyboard connected to the 3DNS Controller, the 3DNS Controller defaults to using the serial port as the console.
3. Connect the primary Ethernet interface (number 12 in Figure 3.2) to the network from which the 3DNS Controller receives connection requests.
4. If you want an additional network connection, connect the secondary Ethernet interface card (number 11 in Figure 3.2) to an alternate network.

5. If you have a hardware-based redundant system, connect the fail-over cable to the terminal serial port on each unit (number 7 in Figure 3.2).
6. Connect the power cable to the 3DNS Controller (number 2 in Figure 3.2), and then connect it to the power source.

WARNING

Before connecting the power cable to a power supply, customers outside the United States should make sure that the voltage selector is set appropriately.

To configure a serial terminal in addition to the console

If you want to configure a serial terminal for the 3DNS Controller in addition to the standard console, you need to follow the configuration steps below. Note that if you are using a serial vt100 connection, you must edit both the **/etc/ttys** and **bash_profile** files on the 3DNS Controller.

1. Configure the serial terminal settings as follows:
  - 9600 baud
  - 8 bits
  - 1 stop bit
  - No parity
2. Open the **/etc/ttys** file and find the line that reads **tty00 off**. Modify it as shown here:

```
# PC COM ports (tty00 is DOS COM1) \  
tty00 "/usr/libexec/getty default" vt100 in  
secure tty01 off
```
3. Save the **/etc/ttys** file and close it.
4. Reboot the 3DNS Controller.

Running the First-Time Boot utility

The First-Time Boot utility is a wizard that walks you through a brief series of required configuration tasks, such as defining a root password, and configuring IP addresses for the network interfaces. Once you complete the First-Time Boot utility, you can connect to the 3DNS Controller from a remote workstation and begin configuring your load balancing set up.

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
- ❖ Interface settings for the network interface(s)
- ❖ Configuration for 3DNS redundant systems (fail-over IP address)
- ❖ IP address for remote administration
- ❖ Settings for the 3DNS web server
- ❖ Defining the sync group
- ❖ Settings for the NameSurfer application
- ❖ Allowance of technical support access

First, you configure all of the required information, then you have the opportunity to confirm each individual setting or correct it if necessary, and then 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.

Gathering the information

Before you run the First-Time Boot utility on a specific 3DNS Controller, you should have the following information ready to enter:

- ❖ Passwords for the root system, for the 3DNS web server, and for technical support access (optional)
- ❖ Host names for the root system and for the 3DNS web server
- ❖ A default route (typically a router's IP address)
- ❖ Settings for the network interfaces, including IP addresses, media type, and custom netmask and broadcast addresses
- ❖ Configuration information for redundant systems, including an IP alias for the shared address, and the IP addresses of the individual controllers
- ❖ The IP address or IP address range for remote administrative connections.
- ❖ The IP addresses of the other 3DNS Controllers and BIG/ip Controllers running in the network.

An important note about configuring international 3DNS Controllers

When you run the First-Time Boot utility on an international 3DNS Controller, certain screens are different from the screens that are shown when you run the First-Time Boot utility on a US 3DNS Controller. On US 3DNS Controllers, the First-Time Boot utility prompts you to configure an administrative IP address from which the 3DNS Controller accepts **ssh** connections. On international 3DNS Controllers, the First-Time Boot utility prompts you to configure an administrative IP address from which the 3DNS Controller accepts **rsh** connections.

The 3DNS Controller stores the administrative IP address for **rsh** and **rcp** connections in the **/etc/hosts.allow** file. Note that storing the administrative IP address in the **/etc/hosts.allow** file may be slightly different from other common **rsh** configurations where it is often stored in the **/etc/hosts.equiv** file.

Starting the First-Time Boot utility

You run the First-Time Boot utility directly on the console, using the VGA monitor and keyboard. Once you turn on the power switch (located on the front of the 3DNS Controller as shown in Figure 3.1, number 7), the 3DNS Controller displays the License Agreement screen. You must scroll through the screen, read it and accept the agreement before you can move to the next screen. If you agree to the license statement, the next screen you see is the Welcome screen. From this screen, simply press any key on the keyboard, and then follow the instructions on the subsequent screens to complete the process.

Defining a root password

A root password allows you administrative access to the 3DNS Controller system. The password must contain a minimum of 6 characters, but no more than 128 characters. Passwords are case-sensitive, and we recommend that your password contain a combination of upper and lowercase characters, as well as punctuation characters. Once you enter a password, the First-Time Boot utility prompts you to confirm your root password by typing it again. If the two passwords match, your password is immediately saved. If the two passwords do not match, you receive an error message and are asked to re-enter your password.

WARNING

---

*The root password is the only setting that is saved immediately, rather than confirmed and committed at the end of the First-Time Boot utility process. You cannot change the root password until the First-Time Boot utility completes and you reboot the 3DNS Controller (see *To change the root user password for command line access*, on page 7-33 for details). Note that you can change other system settings when the First-Time Boot utility prompts you to confirm your configuration settings.*

Defining a host name

The host name identifies the 3DNS Controller itself. Host names must start with a letter or number, and must be at least two characters. They may contain numbers, letters, and the symbols for a dash (-) and a period (.) if you like. There are no additional restrictions on host names, other than those imposed by your own network requirements.

Configuring a default route

If a 3DNS Controller does not have a predefined static route for network traffic, the unit automatically sends traffic to the IP address that you define as the default route. Typically, a default route is set to a router's IP address.

Configuring a time zone

Next, you need to specify your time zone. This ensures that the clock for the 3DNS Controller is set correctly, and that dates and times recorded in log files correspond to the time zone of the system administrator. Scroll through the system file to find the time zone closest to your location. Note that one option may appear with multiple names. Select the time zone you want to use, and press Enter to continue.

Configuring the interfaces

On the Configure 3DNS Interfaces screen, select **Yes** if you have a redundant system. You must configure the primary Ethernet interface, but you configure the secondary Ethernet interface only if you want to have two independent network access paths to the 3DNS Controller. The utility prompts you for each interface, and asks you to provide the IP address, netmask, broadcast address, and the interface media type.

If you have a redundant system, you are also prompted to provide the IP address that serves as an IP alias for both 3DNS Controllers. The IP alias is shared between the units, and is used only by the currently active machine. Each individual controller uses unique IP addresses on its network interface card(s). The First-Time Boot utility guides you through configuring the interfaces, based on your hardware configuration:

❖ **Stand-alone controllers**

On stand-alone controllers you enter IP addresses in the following order: primary Ethernet interface IP address, secondary Ethernet interface IP address.

❖ **Redundant systems**

On redundant systems, you enter IP addresses in the following order: primary Ethernet interface IP address, primary shared alias, secondary Ethernet interface IP address, secondary shared alias.

Configuring the primary Ethernet interface

The Select Interface screen shows a list of the installed interfaces. Select the one you want to use for the primary Ethernet interface, and press Enter (the primary Ethernet interface is typically named **exp0**). The utility prompts you for the following information, in many cases offering you a default:

- ❖ Interface IP address
- ❖ Netmask
- ❖ Broadcast address
- ❖ Primary shared IP alias (redundant systems only)
- ❖ Primary shared alias netmask (redundant systems only)
- ❖ Primary shared alias broadcast address (redundant systems only)
- ❖ Interface media type

- ❖ Peer IP address (redundant systems only)

◆ **Note**

The IP address of the primary Ethernet interface is not the IP address associated with your domain(s). The IP addresses of the domains themselves are specified by the wide IP definitions.

◆ **WARNING**

The First-Time Boot utility lists only the network interface cards that it detects during boot up. If the utility lists only one interface card, the network adapter may have come loose during shipping. Check the LED indicators on the network adapters to ensure that they have properly detected the 3DNS Controller media that should be installed.

Once you select the interface, you need to enter the following information:

- ❖ **IP address**

- ❖ **Netmask**

Note that the 3DNS Controller uses a default netmask appropriate to the subnetwork indicated by the IP address. The default netmask is shown in brackets at the prompt. To accept the default, press Return.

- ❖ **Broadcast address**

The default broadcast address is a combination of the IP address and the netmask. The default broadcast address is shown in brackets at the prompt. To accept the default, press Return.

- ❖ **Primary shared IP alias (redundant systems only)**

- ❖ **Peer IP address (redundant systems only)**

The peer IP address is the IP address of the other controller that runs in the redundant system. The 3DNS Controller uses the specified peer IP address to communicate with the second controller.

❖ **Media type for primary Ethernet interface**

The media type options depend on the network interface card included in your hardware configuration. The 3DNS platform supports the following types:

- auto
- 10baseT
- 10baseT,FDX
- 100baseTX
- 100baseTX,FDX
- FDDI
- Gigabit Ethernet

Configuring a secondary Ethernet interface

If you want to have two separate network access paths to the 3DNS Controller, you can configure a secondary Ethernet interface to connect to the network as well. When you configure the internal network interface, you input the same type of information that you used to configure the primary Ethernet interface:

❖ **IP address**

❖ **Netmask**

Note that the 3DNS Controller uses a default netmask appropriate to the subnetwork indicated by the IP address. The default netmask is shown in brackets at the prompt. To accept the default, press Return.

❖ **Broadcast address**

The default broadcast address is a combination of the IP address and the netmask. The default broadcast address is shown in brackets at the prompt. To accept the default, press Return.

❖ **Secondary shared IP alias (redundant systems only)**

❖ **Peer IP address (redundant systems only)**

The peer IP address is the IP address of the other controller that runs in the redundant system. The 3DNS Controller uses the specified peer IP address to communicate with the second controller.

❖ **Media type for secondary Ethernet interface**

The media type options depend on the network interface card included in your hardware configuration. The 3DNS Controller supports the following types:

- auto
- 10baseT
- 10baseT,FDX
- 100baseTX
- 100baseTX,FDX
- FDDI
- Gigabit Ethernet

Configuring remote administration

The screens that you see for configuring remote administration vary, depending on whether you have a US 3DNS Controller, or an international 3DNS Controller. On a US 3DNS Controller, the first screen you see is the Configure SSH screen, which prompts you to type in an address for SSH command line access. On international 3DNS Controllers that do not have SSH, the First-Time Boot utility skips this screen (international 3DNS Controllers host remote administrative connections using **rsh**).

Next, the First-Time Boot utility prompts you to enter a single IP address or a range of IP addresses, from which the 3DNS Controller will accept administrative connections (either remote shell connections, or connections to the 3DNS web server). To specify a range of IP addresses, you can use the asterisk (*) as a wildcard character in the IP addresses.

The following example allows remote administration from all hosts on the **192.168.2.0** network:

192.168.2.*

◆ **Tip**

You must configure command line access for redundant systems. If you do not configure command line access, the two controllers in the system cannot communicate with each other, and they cannot properly initiate a fail-over.

Configuring settings for the 3DNS web server

The 3DNS web server requires you to define a domain name for the server on the primary Ethernet interface, and, if you are using the secondary Ethernet interface, you must define a domain name on that interface as well. The 3DNS web server configuration requires that you define a user ID and password. On US products, the configuration also generates certificates for authentication.

The First-Time Boot utility guides you through a series of screens to set up web server access:

- ❖ The first screen prompts you to enter a fully qualified domain name for each network interface that you configured earlier in the utility.
- ❖ The certification screen prompts you to enter the company name and location information used for the authentication certificate (US 3DNS Controllers only).
- ❖ The next web server screen prompts you for a user name and a password. The password does not show on screen as you type it, and the utility asks you to enter it again for confirmation purposes.
- ❖ The final screen prompts to enter whether you want to allow F5 technical support to have access to the web server.

Note that if you ever change the IP addresses or host names on the 3DNS Controller interfaces, you need to reconfigure the 3DNS web server to reflect your new settings. You can run the re-configuration utility from the command line using the following command:

config-httpd

If you wish to create a new password for the 3DNS web server, delete the **/var/f5/httpd/basicauth/users** file before running the **config-httpd** utility. If this file is missing from the configuration, the utility prompts you for both user ID and password information.

You can also add users to the existing password file, change a password for an existing user, or recreate the password file, without actually going through the 3DNS web server configuration process. For more information, see *To add a new user ID using the F5 Configuration utility*, on page 7-33.

WARNING

*If you have modified the 3DNS web server configuration outside of the configuration utility, be aware that some changes may be lost when you run the **config-httpd** utility. This utility overwrites the **httpd.conf** file, and several other files, but it does warn you before doing so.*

Identifying additional controllers in the network

In the next series of screens, you identify other 3DNS Controllers and BIG/ip Controllers running on the network that the current 3DNS Controller needs to communicate with. For each additional 3DNS Controller or BIG/ip Controller, you need to enter the IP address, the root user ID, whether you access the controller using SSH or **rsh**, and you need to select which sync group the controller is a part of. A **sync group** is a group of 3DNS Controllers that share configuration settings and path statistics.

Note that if you are defining a redundant system (either 3DNS Controllers or BIG/ip Controllers), you need to enter the IP address of controller, as well as the shared IP alias for each interface on the controller. You also need to choose whether the current 3DNS

Controller will be a principal or a receiver (note that each sync group can have only one principal, but can have an unlimited number of receivers).

At any time, you can view the list of currently defined controllers, and you can use the **List is Complete** option to exit these screens and move on to the next configuration task.

Configuring the NameSurfer application for zone file management

In the final series of screens, you chose whether you want to use NameSurfer to handle DNS zone file management on the current 3DNS Controller. We strongly recommend that you configure NameSurfer to handle zone file management by selecting NameSurfer to be the master on the 3DNS Controller. After you select NameSurfer as the master, NameSurfer converts the master DNS zone files on the controller and handles all changes and updates to these files (note that you can access the NameSurfer application directly from the F5 Configuration utility). For details about converting existing BIND files to NameSurfer, see *To transfer and convert existing BIND files* on page C-21.

WARNING

If you do not set NameSurfer to be the master for your wide IP zones, you cannot use the F5 Configuration utility. Instead, you must manually configure all 3DNS Controller settings.

Confirming your configuration settings

At this point, you should have entered all the configuration information, and now you confirm each setting. Each confirmation screen displays a setting, and prompts you to accept or re-enter it. If you choose to edit it, the utility displays the original configuration screen in which you defined the setting the first time. When you finish editing the item, you return directly to the

Confirmation screen for that item, and continue the confirmation process. Note that once you accept a setting in the Confirmation screen, you do not have another opportunity to review it.

You confirm or edit the settings in the same order that you configured them:

- ❖ Confirm Host name
- ❖ Confirm Default route
- ❖ Confirm time zone
- ❖ Confirm all interface settings, external and internal
- ❖ Confirm administrative IP address
- ❖ Confirm web server options

Once you have confirmed the last setting, the First-Time Boot utility moves directly into the commit phase, where you are not able to make any changes.

Committing your configuration settings to the system

Once you confirm all of the configuration settings, the configuration utility saves the configuration settings. During this commit process, the First-Time Boot utility creates the following files and tables:

- ❖ An administrative IP access file
This file stores the IP address, or IP address range, from which the 3DNS Controller accepts administrative connections.
- ❖ An **/etc/wideip.conf** file
- ❖ An **/etc/netstart** file
- ❖ An **/etc/ethers** file
- ❖ A **/var/f5/httpd/conf/httpd.conf** file
- ❖ An **/etc/sshd_config** file

If you want to change any information in these files at a later time, you can edit the files directly, you can change the information in the web-based Configuration utility, or for certain settings, you can change them using command line utilities. If necessary, you can also re-run the First-Time Boot utility.

Enabling remote login tools

If you are setting up a US 3DNS Controller that needs to communicate with international 3DNS Controllers, you need to enable the **rsh** and **rcp** tools on the US 3DNS Controller. These are the standard communication and copying tools that international 3DNS Controller and BIG/ip Controllers use.

To enable the remote login tools on a US 3DNS Controller

Run the **rsetup** script from the command line. The **rsetup** script performs several essential steps to enable access for **rsh** and **rcp**, and we strongly recommend that you use the script rather than doing this manually.

Preparing workstations for command line access

The type of system you have determines the options you have for remote command line administration:

- ❖ 3DNS Controllers distributed in the US support secure shell command line access using the F-Secure SSH client.
- ❖ 3DNS Controllers distributed outside the US support command line access using a standard **rsh** shell.

If you are working with a US 3DNS Controller, you probably want to install the F-Secure SSH client on your workstation. The 3DNS Controller includes a version of the F-Secure SSH client for each of the following platforms: Windows, UNIX, and Macintosh. You can download the F-Secure client using your web browser, or you can download the client using an FTP server on the administrative workstation.

Note that the F-Secure license agreement allows you to download two copies of the F-Secure SSH client. If you require additional licenses, you need to contact Data Fellows. For information about

contacting Data Fellows, as well as information about working with the SSH client, refer to the F-Secure manual included with your 3DNS Controller.

◆ **Note**

You can also use the F-Secure SSH suite for file transfer to and from the 3DNS Controller, as well as for remote backups. An F-Secure SSH client is pre-installed on the 3DNS Controller to assist with file transfer activities. Please refer to the F-Secure User's Manual for more information.

Downloading the F-Secure SSH client from the 3DNS web server

The F-Secure SSH client is available in the Downloads section of the 3DNS web server. For US products, you connect to the 3DNS web server via SSL on port 443 (use **https://** rather than **http://** in the URL). Once you connect to the 3DNS web server, click the **Downloads** link. From the Downloads page, you can select the SSH Client.

Downloading the F-Secure SSH client using FTP

The 3DNS Controller has an FTP client installed, which allows you to transfer the F-Secure SSH Client using FTP (note that your destination workstation must also have an FTP server installed). After you transfer the installation file, you simply decompress the file and run the F-Secure installation program.

You initiate the transfer from the 3DNS Controller itself, using the monitor and keyboard, or the serial terminal, attached directly to the 3DNS Controller.

To transfer the SSH client using FTP

1. Locate the SSH client that is appropriate for the operating system that runs on the administrative workstation:

- a) Go to the **/usr/contrib/fsecure** directory where the F-secure SSH clients are stored.
- b) List the directory, noting the file name that corresponds to the operating system of your administration workstation.
3. Start FTP:
ftp
4. Open a connection to the remote workstation using the following command, where **IP address** is the IP address of the remote workstation itself:
open <IP address>

Once you connect to the administrative workstation, the FTP server on the administrative workstation prompts you for a password.
5. Enter the appropriate user name and password to complete the connection.
6. Switch to passive FTP mode:
passive
7. Switch the transfer mode to binary:
bin
8. Go to the directory on the administrative workstation where you want to install the F-Secure SSH client.
9. Start the transfer process using the following command, where **filename** is the name of the F-Secure file that is specific to the operating system running on the administrative workstation:
put <filename>
10. Once the transfer is done, type the following command:
quit

Setting up the F-Secure SSH client on a Windows 95 or Windows NT workstation

The F-Secure SSH client installation file for Windows platforms is compressed in ZIP format. You can use standard ZIP tools, such as PKZip or WinZip to extract the file.

To unzip and install the SSH client

1. Log on to the Windows workstation.
2. Go to the directory to which you transferred the F-Secure installation file. Run PKZip or WinZip to extract the files.
3. The set of files extracted includes a Setup executable. Run the Setup executable and install the client.
4. Start the F-Secure SSH client.
5. In the SSH Client window, from the File menu choose **Connect**.
The Connect Using Password Authentication window opens.
6. Click **Properties**.
7. In the Options dialog box, check **Compression** and **Forward X11**, and set the Cipher option to **Blowfish**. Click OK to return to the Connect Using Password Authentication window.
8. In the Connect Using Password Authentication window, type the following items:
  - a) 3DNS Controller IP address or host name
  - b) The root user name
  - c) The root password
9. Press the Return key to log on to the 3DNS Controller.

Setting up the F-Secure SSH client on a UNIX workstation

The F-Secure installation file for UNIX platforms is compressed in TAR/Gzip format.

To untar and install the SSH client

1. Log on to the workstation and go to the directory into which you transferred the F-Secure SSH client tar file.
2. Untar the file and follow the instructions in the **install** file to build the F-Secure SSH client for your workstation.
3. Start the SSH client.
4. Open a connection to the 3DNS Controller:
ssh -l root [3DNS IP address]
5. Enter the root password.

4

Defining the Network Setup

- **Setting up a basic configuration**
- **Setting up a data center**
- **Setting up servers**
- **Setting up sync groups**
- **Configuring global variables**
- **Configuring IP filters**
- **Configuring Sendmail**

Setting up a basic configuration

The second phase of installing 3DNS Controllers is to define the network setup. Each 3DNS Controller in the network setup needs to understand which data center stores specific servers, and which other 3DNS Controllers it can share configuration and load balancing information with. A basic network setup includes data centers, servers, wide IPs, and one sync group.

In addition, you can also configure global variables that apply to all servers and wide IPs in your network. Because the default values work well for most situations, configuring global variables is entirely optional, and you can find more information about it in *Configuring global variables*, on page 4-29.

❖ Data centers

Data centers are the top level of your network setup. We recommend that you configure one data center for each physical location in your global network. A data center defines the servers (3DNS Controllers, BIG/ip Controllers, and hosts) that reside at that location.

Each data center can contain any type of server. For example, in Figure 4.1, the Tokyo data center contains a 3DNS Controller and a host, while the New York and Los Angeles data centers contain 3DNS Controllers and BIG/ip Controllers.

To configure data centers, see *Setting up a data center*, on page 4-2.

❖ Servers

The servers that you define in the network setup include 3DNS Controllers, BIG/ip Controllers, and host machines. You need to configure the 3DNS Controllers that manage the BIG/ip Controllers and hosts, as well as the virtual servers that are managed by the BIG/ip Controllers and hosts. Virtual servers are the ultimate destination for connection requests.

To configure servers, see *Setting up servers*, on page 4-5.

❖ **Sync groups**

Sync groups contain only 3DNS Controllers. By setting up a sync group, you define which 3DNS Controllers have the same configuration. In most cases, you should define all 3DNS Controllers as part of the same sync group.

To configure sync groups, see *Setting up sync groups*, on page 4-27.

❖ **Wide IPs**

After you configure virtual servers on your BIG/ip Controllers and hosts, you need to specify how connections are distributed among the virtual servers by defining wide IPs. A wide IP maps a domain name to a pool of virtual servers, and it specifies the load balancing modes that the 3DNS Controller uses to choose a virtual server from the pool. When a local DNS requests a connection to a specific domain name, the wide IP definition specifies which virtual servers are eligible to answer the request, as well as which load balancing modes to use in choosing one virtual server for resolving each request.

To configure wide IPs, see *Defining a wide IP*, on page 5-2.

Setting up a data center

The first step in configuring your 3DNS Controller network is to create data centers. A data center defines the group of 3DNS Controllers, BIG/ip Controllers, and hosts that reside in a single physical location.

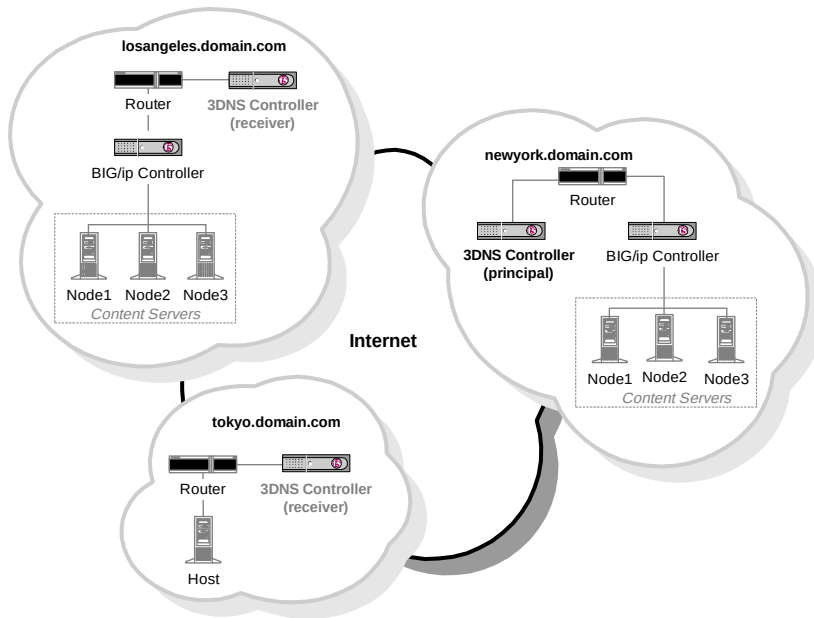


Figure 4.1 Example network setup

The advantage of grouping all machines in one location into one data center is that, by doing so, you are allowing path information collected by one machine to be shared with all other machines in the data center. For example, when a host machine belongs to a data center, the host can take advantage of the information collected by the **big3d** agent, which only runs on 3DNS Controllers and BIG/ip Controllers. Without the information that the **big3d** agent collects, you would not be able to use dynamic load balancing modes for virtual servers owned by host machines.

To configure a data center using the F5 Configuration utility

1. In the navigation pane, click **Data Centers**.

2. On the toolbar, click **Add Datacenter**.
The Add New Datacenter screen opens.
3. In the **Name** box, type a name for the data center.
4. In the **Location** box, type the physical location of the data center (optional).
5. In the **Contact** box, type the name of the data center administrator or the name of the department that manages the data center.

Although this information is not required, it can be useful if problems later arise or changes are required.

6. Click **Add**.

The data center is added to your configuration. Repeat this process for each data center in your network. When you add servers to the network setup, you will assign the servers to the appropriate data centers.

To configure a data center manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
An environment variable determines whether this command starts vi or pico.
2. Locate or add the **datacenter** statement.

The **datacenter** statement should be the second statement in the file, after the **globals** statement and before **server** statements.
3. In the first line of the **datacenter** statement, type a name for the data center and enclose the name in quotation marks, as shown in Figure 4.2.
4. Enter the server type and IP address for each server that is part of the specified data center.

Figure 4.2 shows the correct syntax for the **datacenter** statement:

```
datacenter {  
  name <"data center name">  
  [ location <"location info"> ]  
  [ contact <"contact info"> ]  
  [ 3dns <3DNS IP address> ]  
  [ bigip <BIG/ip IP address> ]  
  [ host <host IP address> ]  
}
```

Figure 4.2 Syntax for the datacenter statement

Repeat the above procedure until you have added a separate **datacenter** statement for each data center on your network.

Figure 4.3 shows a sample **datacenter** statement:

```
datacenter {  
  name "New York"  
  location "NYC"  
  contact "3DNS_Admin"  
  3dns 192.168.101.2  
  bigip 192.168.101.40  
  host 192.168.105.40  
}
```

Figure 4.3 Sample data center definition

Setting up servers

There are three types of servers: 3DNS Controllers, BIG/ip Controllers, and other hosts. At the very minimum, your network includes at least two servers: one 3DNS Controller and one server (BIG/ip Controller or host) that it manages.

This section describes how to set up each 3DNS Controller, BIG/ip Controller, and host machine that make up your network. The setup procedures here assume that the BIG/ip Controllers and hosts

are up and running, and that they already have virtual servers defined. Note that 3DNS Controllers do not manage virtual servers.

Defining 3DNS Controller servers

The purpose of defining a 3DNS Controller server is to establish where the 3DNS Controller resides (in which data center) and to change **big3d** agent settings if you prefer. In setting up a 3DNS Controller server, you also make that 3DNS Controller available to be added to a sync group.

To define a 3DNS Controller server using the F5 Configuration utility

1. In the navigation pane, click **3DNS Servers**.
2. On the toolbar, click **Add 3DNS Server**.
The Add New 3DNS Server screen opens.
3. In the **3DNS Server Name** box, type the name of the 3DNS Controller.
4. In the **3DNS Server IP Address** box, type the IP address of the 3DNS Controller. If the server is actually a redundant system, type the shared IP alias.
5. Check the **Secure** option to specify that the 3DNS Controller uses **ssh** (secure shell) for remote connections. The default setting for US 3DNS Controllers is to use **ssh**. The default setting for international 3DNS Controllers is to use **rsh**, an uninterrupted shell.
6. In the **User** box, specify the user name that you want to use for remote logins. Enclose this name in quotation marks. If you omit this parameter, the default, "**root**", is used.
7. If you want to turn the **big3d** agent off, clear the check boxes next to each factory.

8. If you want to change the number of factories that the **big3d** agent runs, type the new value in the corresponding factory box (you can specify separate numbers for each factory).
9. If the server is actually a redundant system, type the IP address of each unit in the system in the **Interface Settings** boxes.
10. Click **Next**.
The Data Centers screen opens.
11. Select the data center where the 3DNS Controller is located, and click **Finish**.

The 3DNS Controller is added to your configuration. Repeat this procedure for each 3DNS Controller you need to add.

To define a 3DNS Controller server manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Use the syntax shown in Figure 4.4 to define a 3DNS Controller.

All **server** statements should appear after the **sync_group** statement and before **wideip** statements.

```
server {
  type 3dns
  address <IP address>
  name <"3dns_name">
  [ remote {
    secure <yes | no>
    user <"user name">
  } ]
  [ interface {
    address <NIC IP address>
    address <NIC IP address>
  } ]
  [ factories {
    prober <number>
    discovery <number>
    snmp <number>
    hops <number>
  } ]
  [ prober <IP address> ]
  probe_protocol < icmp | udp | tcp >
  port <port to probe>
}
```

Figure 4.4 Syntax for defining a 3DNS Controller server

Figure 4.5 shows a sample **server** statement that defines a 3DNS Controller:

```
// New York
server {
    type 3dns
    address 192.168.101.2
    name "3dns-newyork"
    remote {
        secure no
        user "root"
    }
    prober 192.168.101.40
    probe_protocol icmp
    port 53
}
```

Figure 4.5 Sample 3DNS Controller server definition

Defining BIG/ip Controller servers

Before you define BIG/ip Controller servers, you need to have the following information:

- ❖ The IP address and service name or port number of each virtual server to be managed by the BIG/ip Controller.
- ❖ The IP address of the server itself.

To define a BIG/ip Controller server using the F5 Configuration utility

1. In the navigation pane, click **BIG/ips**.
2. On the toolbar, click **Add BIG/ip**.
The Add BIG/ip Server screen opens.
3. In the **BIG/ip Server Name** box, type the name of the BIG/ip Controller.
4. In the **BIG/ip IP Address** box, type the IP address of the BIG/ip Controller. If the server is actually a redundant system, type the shared IP alias.

5. Check the **Secure** option to specify that the BIG/ip Controller uses **ssh** (secure shell) for remote connections. The default setting for US BIG/ip Controllers is to use **ssh**. The default setting for international BIG/ip Controllers is to use **rsh**, an unencrypted shell.
6. In the **User** box, specify the user name that you want to use for remote logins. Enclose this name in quotation marks. If you omit this parameter, the default, "**root**", is used.
7. If you want to turn the **big3d** agent off, clear the check boxes next to each factory.
8. If you want to change the number of factories that the **big3d** agent runs, type the new value in the corresponding factory box (you can specify separate numbers for each factory).
9. If the server is actually a redundant system, type the IP address of each unit in the system in the **Interface Settings** boxes.
10. Click **Next**.
The Data Centers screen opens.
11. Select the data center where the BIG/ip Controller is located, and click **Next**.
The Configure Virtual Server screen opens.
12. In the **Virtual Server IP Address** box, type the virtual server's IP address.
13. In the **Virtual Server Port** box, type the virtual server's port number or select a service from the list.
14. To allow iQuery packets to pass through firewalls, complete the **Translate IP Address** and **Translate Port** boxes. See *Setting up iQuery communications for the big3d agent*, on page 2-20 for details.
15. Click **Finish**.

The BIG/ip Controller and the specified virtual server are added to your configuration.

To add additional virtual servers using the F5 Configuration utility

1. In the navigation pane, click **BIG/ips**.
2. In the **BIG/ip Virtual Servers** column, click the BIG/ip Controller that you just added.
3. On the toolbar, click **Add Virtual Servers**.
4. In the **Virtual Server IP Address** box, type the virtual server's IP address.
5. In the **Virtual Server Port** box, type the virtual server's port number or select a service from the list.
6. To allow iQuery packets to pass through firewalls, complete the **Translate IP Address** and **Translate Port** boxes. See *Setting up iQuery communications for the big3d agent*, on page 2-20 for details.
7. Click **Add**.

Repeat this process for each virtual server you want to add to this BIG/ip Controller.

To define a BIG/ip Controller server manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Use the syntax shown in Figure 4.6 to define a BIG/ip Controller.

All **server** statements should appear after the **sync_group** statement and before **wideip** statements.

If you need to allow iQuery packets to pass through firewalls, include the **translate** keyword in the **server** statement that defines the BIG/ip Controller. When you include the **translate** keyword, the iQuery utility includes translated IP addresses in the packets sent to the specific BIG/ip Controller. See *Setting up iQuery communications for the big3d agent*, on page 2-20 for details.

```
server {
  type bigip
  address <IP address>
  name <"bigip_name">
  [ remote {
    secure <yes | no>
    user <"user name">
  } ]
  [ interface {
    address <NIC IP address>
    address <NIC IP address>
  } ]
  [ factories {
    prober <number>
    discovery <number>
    snmp <number>
    hops <number>
  } ]

  vs {
    address <virtual server IP address>
    port <port number> | service <"service name">
    [ translate {
      address <IP address>
      port <port number>|service <"service name">
    } ]
  }
}
```

Figure 4.6 Syntax for defining a BIG/ip Controller server

Figure 4.7 shows a sample **server** statement that defines a BIG/ip Controller:

```
server {
    type                bigip
    address              192.168.101.40
    name                "bigip-newyork"
    remote {
        secure          yes
        user             "administrator"
    }
    # Tell 3DNS about the 2 interfaces on a BIG/ip HA
    interface {
        address          192.168.101.41
        address          192.168.101.42
    }
    # Change the number of factories doing the work at big3d
    factories {
        prober           6
        discovery        1
        snmp              1
        hops              2
    }
    vs {
        address          192.168.101.50
        service           "http"
        translate {
            address       10.0.0.50
            port          80
        }
    }
    vs {
        address          192.168.101.50:25 // smtp
        translate {
            address       10.0.0.50:25
        }
    }
}
```

Figure 4.7 Sample BIG/ip Controller server definition

Defining host servers

A host is an individual network server or server array controller other than the BIG/ip Controller. Before you configure a host, you should know the following:

❖ **Address information**

Collect the IP address and service name or port number of each virtual server to be managed by the host.

❖ **SNMP information for host probing**

If you want to implement host probing, you need to specify SNMP agent settings after you define the host server. The settings you specify include the type and version of SNMP agent that runs on the host, the community string, and the number of communication attempts that you want the big3d agent to make while gathering host metrics. SNMP agent settings for hosts are described in *Configuring host SNMP settings*, on page 4-17.

◆ **Note**

---

*To fully configure host probing, you need to configure the SNMP agent settings in the host definition as previously described, but you also need to set up the **big3d** agents to run SNMP factories, and you need to configure the SNMP agents on the hosts themselves. See *Setting up SNMP probing for hosts*, on page 2-14 for details.*

To define a host server using the F5 Configuration utility

1. In the navigation pane, click **Hosts**.
2. On the toolbar, click **Add Host IP**.
3. In the **Host Name** box, enter the name of the host.
4. In the **Host IP Address** box, type the IP address of the host.
5. In the **Host Port** box, type the host's port number or select a service from the list.
6. If you want a **big3d** agent to collect path data on behalf of the host, type the IP address of a 3DNS Controller or BIG/ip Controller that runs the **big3d** agent in the **Prober**

IP Address box. Note that the controller that runs the **big3d** agent must be located in the same data center as the host.

7. If you specify a prober IP address, specify whether the prober uses the ICMP, TCP, or UDP protocol in the **Probe Protocol** box.
8. Click **Next**.
The Data Centers screen opens.
9. Select the data center where the host is located, and click **Next**.
The Configure Virtual Server screen opens.
10. In the **IP Address** box, type the virtual server's IP address.
11. In the **Port** box, type the virtual server's port number or select a service from the list.
12. If you are using the Ratio load balancing mode, specify a ratio weight in the **Ratio** box. The ratio weight determines proportionally how often a virtual server will be chosen. The default ratio weight is **1**.
13. Click **Finish**.

The host and the specified virtual server are added to your configuration.

To add additional virtual servers using the F5 Configuration utility

1. In the navigation pane, click **Hosts**.
2. In the table, find the host that you just added.
3. Click the entry in its **Host Virtual Servers** column.
4. On the toolbar, click **Add Host Virtual Servers**.
5. Enter the virtual server's IP address, port, and ratio values as shown in the preceding procedure.
6. Click **Add**.

Repeat this process for each virtual server you want to add to this host.

To define a host server manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Use the syntax shown in Figure 4.8 to define a host.

All **server** statements should appear after the **sync_group** statement and before **wideip** statements.

```
server {
    type host
    address <IP address>
    name <"host_name">
    [ prober <ip_address> ]
    probe_protocol <tcp | icmp | udp>
    port <port number> | service <"service name">
    [ snmp {
        agent <generic | ucd | solstice | ntserv>
        port <port number>
        community <"community string">
        timeout <seconds>
        retries <number>
        version <SNMP version>
    } ]
    vs {
        address <virtual server IP address>
        port <port number> | service <"service name">
        [ probe_protocol <tcp | icmp | udp> ]
    }
}
```

Figure 4.8 Syntax for defining a host server

Figure 4.9 shows a sample **server** statement that defines a host:

```
server {
  type          host
  address       192.168.104.40
  name          "host-tokyo"
  prober        192.168.101.40
  probe_protocol icmp
  port          53
  snmp {
    agent        ucd
    community    "public"
    version      1
  }
  vs {
    address      192.168.104.50:25
  }
  vs {
    address      192.168.104.50:80
  }
}
```

Figure 4.9 Sample host server definition

Configuring host SNMP settings

After defining a host server, you need to configure its SNMP settings if you want to use SNMP host probing. Remember that you must first set up at least one SNMP probing factory on each 3DNS Controller and BIG/ip Controller that runs the **big3d** agent.

Although the 3DNS Controller does not use all of the following information for load balancing, the SNMP prober collects it and the Hosts screen in the Configuration utility displays it for your convenience.

- ❖ Memory utilization
- ❖ CPU utilization
- ❖ Disk space utilization
- ❖ Bytes in/out
- ❖ Packet rate

Of these metrics, the 3DNS Controller uses only the packets in/out metrics to make load balancing decisions.

The 3DNS Controller supports the following host SNMP agents:

❖ **Generic**

The 3DNS Controller can work with a generic SNMP agent running on a host.

❖ **UCD SNMPD**

The UCD SNMPD is a free SNMP agent provided by the University of California at Davis. It is freely available on the web at <http://ucd-snmp.ucdavis.edu>, <ftp://ucd-snmp.ucdomain.edu/ucd-snmp.tar.gz>.

❖ **Solstice Enterprise**

The Solstice Enterprise agent is a product of SunSoft.

❖ **Windows NT 4.0 SNMP**

The Windows NT 4.0 SNMP matrix agent is distributed with the Microsoft Windows NT 4.0 server.

Configuring SNMP agents on hosts, on page 4-20, provides some useful tips for configuring the different SNMP agents on the hosts themselves. We recommend that you use the information in conjunction with the documentation originally provided with the SNMP agent.

To configure host SNMP settings using the F5 Configuration utility

1. In the navigation pane, click **Hosts**.
2. Select a host server from the list.
3. On the toolbar, click **SNMP Configuration**.
The Host SNMP Configuration screen opens.
4. Check the **SNMP Enabled** box.
5. In the **SNMP Port** box, type the port number that the SNMP agent runs on, or select a service from the list.
6. Specify the type of SNMP agent you have in the **Type** box.
7. In the **Community** box, specify the password to use for basic SNMP security and for grouping SNMP hosts.

8. Type the SNMP agent version number in the **Version** box.
9. Accept the defaults in the **Retries** and **Timeout** boxes.

The defaults are appropriate in most cases. If you are contacting a host through a very slow network, you can try increasing both values to improve performance. However, the problem with increasing these values is that a host that is down can hang up the SNMP for an excessive amount of time.

10. Click **Update** to save these settings.

To configure host SNMP settings manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Locate or add the host **server** statement.

All **server** statements should appear after the **sync_group** statement and before **wideip** statements.
3. Define the server type, address, name, prober, probe protocol, and port information as usual.
4. Add the **snmp** statement. Figure 4.10 shows the SNMP syntax in bold.
5. Define the virtual server information as usual.

```
server {
  type host
  address <IP address>
  name <"host_name">
  probe_protocol <tcp | icmp>
  [ prober <IP address> ]
  port <port number> | service <"service name">
  [ snmp {
    agent <generic | ucd | solstice | ntserve>
    port <port number>
    community <"community string">
    timeout <seconds>
    retries <number>
    version <SNMP version>
  } ]
  vs {
    address <virtual server IP address>
    port <port number> | service <"service name">
    [ probe_protocol <tcp | icmp> ]
  }
}
```

Figure 4.10 Configuring host SNMP settings

Configuring SNMP agents on hosts

For host probing to work, you need to verify that the SNMP agent is properly configured on the host. The following sections offer some tips and hints on configuring each type of supported SNMP agent, but you may want to refer to the documentation provided with your SNMP software for more complete configuration information.

Configuring the UCD SNMP agent on the host

The UCD SNMP agent runs on HP-UX, Ultrix, Solaris, SunOS, OSF, NetBSD, FreeBSD, BSDi, Linux, AIX, OpenBSD, Irix, Windows 95, and Windows NT. Please refer to the **ucdFAQ.txt** file for details. On UNIX and UNIX-like systems, the default

location for the configuration and MIB files is in the **/usr/share/snmp** directory. You can find help on **snmpd** options in the **snmpd** man page.

Figure 4.11 shows a sample configuration file in **/usr/share/snmp/snmpd.conf**. This file configures the SNMP agent to define a community. Our example uses **3dnspwd** as the community, which is retrieved from the address **192.168.254.4** using the prober at **192.168.254.240**. It allows read access of the entire SNMP MIB tree, but does not allow write access.

```
-----begin /usr/share/snmp/snmpd.conf-----
#
# To allow write access to the 'system' subgroup from the local
# network with the community string "sysadmin":
#
# - amend the "source" address in the com2sec section
# to match your local network address
# - uncomment the "access admin" line below
#
# You are also strongly advised to change the community string
# to something other than "sysadmin"
# sec.name source community
com2sec local localhost private
com2sec 3dns 192.168.254.240/32 3dnspwd
# sec.model sec.name
group local any local
group public any public
group 3dns group any 3dns
# incl/excl subtree mask
view all included .1 80
view system included system fe
view mib2 included .iso.org.dod.internet.mgmt.mib-2 fc
# context sec.model sec.level prefix read write not
#access admin "" any noauth 0 mib2 system none
access public "" any noauth 0 system none none
access local "" any noauth 0 all all all
access 3dns group "" any noauth 0 all none none
-----eof /usr/share/snmp/snmpd.conf-----
```

Figure 4.11 Configuring a UCD SNMP agent on the host

Figure 4.12 shows the corresponding host **server** statement:

```
server {
    type host
    address 192.168.254.4 # address of host + SNMP agent
    prober 192.168.254.240 # SNMP prober reader
    snmp {
        agent ucd
        community 3dnspwd
    }
    vs {
        address 192.168.254.201
    }
    : : :
}
```

Figure 4.12 *Configuring the host server statement to run the UCD SNMP agent*

Configuring the Solstice SNMP agent on the host

The Solaris or SunOS 5.x should include the Solstice Master Agent in the distribution CD. The following is a sample configuration that should work for host probing.


```
-----begin /etc/snmp/conf/snmpd.conf-----
# Copyright 1988-01/28/97 Sun Microsystems, Inc. All Rights Reserved.
#pragma ident "@(#)snmpd.conf 2.22 97/01/28 Sun Microsystems"
# See below for file format and supported keywords
sysdescr Sun SNMP Agent,
syscontact System administrator
sysLocation System administrators office
#
system-group-read-community public
#system-group-write-community private
#
read-community public
#write-community private
#
trap localhost
trap-community SNMP-trap
#
#kernel-file /vmunix
#
#managers 192.168.254.240
#####
# File Format:
# Each entry consists of a keyword followed by a parameter
# string, terminated by a newline. The keyword must begin in the
# first position. The parameters are separated from the keyword
# (and from one another) by whitespace. All text following (and
# including) a '#' character is ignored. Case in keywords is
# ignored, but case in parameter strings is NOT ignored.
```

Figure 4.13 Configuring a Solstice SNMP agent on the host (continued on next page)

```

# Supported Keywords:
# sysdescr String to use for sysDescr.
# syscontact String to use for sysContact.
# syslocation String to use for sysLocation.
# system-group-read-community Community name needed for read
# access to the system group.
# system-group-write-community Community name needed for write
# access to the system group.
# read-community Community name needed for read access
# to the entire MIB.
# write-community Community name needed for write access
# to the entire MIB (implies read access).
#
# trap Host names where traps should be sent.
# A maximum of 5 hosts may be listed.
# trap-community Community name to be used in traps.
#
# kernel-file Filename to use for kernel symbols.
#
# managers Hosts that can send SNMP queries.
# Only five hosts may be listed on any one line.
# This keyword may be repeated for a total of 32 hosts.
#
# newdevice Additional devices which are not built in snmpd
# format as below
#
# newdevice type speed name
#
# where newdevice is keyword, type is an interger which has to
# match yourschema file, speed is the new device's speed, and
# name is this newdevice's name
-----eof /etc/snmp/conf/snmpd.conf-----

```

Figure 4.14 *Configuring a Solstice SNMP agent on the host (continued from previous page)*

This allows **192.168.254.240** to query the Solstice SNMP agent. And, its community is public. The **wideip.conf** would be similar to the example for UCD except that the community is "public".

Configuring the Windows NT 4.0 SNMP agent on the host

To configure the Windows NT 4.0 SNMP agent, you need to complete the following tasks.

Install the SNMP agent via the Network Services

1. Right-click the Network Neighborhood icon on your desktop.
2. From the popup menu, select **Properties**.
3. In the Properties dialog box, click the **Services** tab.
4. Click **Add**, and then choose the SNMP service from the service list.
5. Configure community name, IP address allowed to query, etc., so as to reflect the same configuration as specified in the **wideip.conf** file.

Note that whatever service pack you have previously installed into your Windows NT server you must reinstall in order for the SNMP agent to work.

Configure the SNMP server

When you configure the SNMP server, you need to provide the contact, community, and permission information that allows the **big3d** agent to read the SNMP MIB. Note that you cannot change the SNMP configuration when the SNMP service is running. You can temporarily stop the SNMP service by typing **net stop snmp** at the command prompt, and you can restart the service by typing **net start snmp** when you are finished making configuration changes.

Install the Windows NT Resource Kit

If you are doing a typical setup, you should install the Windows NT Resource Kit if it is not already installed on the server. These utilities should provide you with the following important files:

- ❖ **MIBCC.EXE** (MIB compiler)
- ❖ **SNMPMON.EXE** (SNMP monitor)
- ❖ **SNMPUTIL.EXE** (**get/walk/getnext** utility)

❖ **PERF2MIB.EXE**

❖ **LMMIB2.MIB**

❖ **MIB_II.MIB**

❖ **SMI.MIB**

Verify that the SNMP server is running

1. Go to the **Services** tab to make sure the SNMP server is up and running.
2. From the directory where you installed the resource kit utilities, run the following at the command prompt:

c:\utilities\perfm

The **perfm.bat** file effectively creates the performance monitoring agent's **.dll**, automatically loads it, and then restarts the SNMP agent.

Verify the installation

To verify that the Windows NT SNMP is working, go to the 3DNS Controller or BIG/ip Controller that runs the **big3d** SNMP factory. Run either the **snmptest** or **snmpwalk** commands.

◆ Note

*Before running **snmptest** or **snmpwalk** be sure that the ephemeral ports are open by typing the command:*

sysctl -w bigip.open_3dns_lockdown_ports=1

◆ WARNING

We strongly recommend that you do not run a screensaver on your Windows NT server when it is running an SNMP agent. If you run a screensaver and the SNMP agent simultaneously, the CPU utilization reported by NT may show as 100% busy.

Setting up sync groups

A sync group defines the group of 3DNS Controllers that synchronize their configuration settings and metrics data. You configure a sync group from the principal 3DNS Controller. First list the IP address of the principal itself. Then list all other 3DNS Controllers, in the order that they should become principals should previously listed 3DNS Controllers fail.

Each 3DNS Controller in your network must be included in a sync group. There may be cases where you do not want a 3DNS Controller to share its configuration with other controllers. In this case, you can create a separate sync group for each 3DNS Controller. Each sync group would only contain its own name or IP address.

```
sync_group {
  name "sync-ny"
  3dns 192.168.101.2    // New York
}

sync_group {
  name "sync-la"
  3dns 192.168.102.2    // Los Angeles
}
```

Figure 4.15 Sample non-syncing *sync groups* statements

◆ Note

*To implement such a configuration, you must modify each 3DNS Controller's **wideip.conf** file; the F5 Configuration utility does not support this function.*

To define a sync group using the F5 Configuration utility

1. In the navigation pane, click **3DNS Sync**.
The System - Synchronization screen opens.

2. In the **Sync Group** box, type the name of the new sync group and click **Add Group**.
3. On the toolbar, click **Add to Group**.
The Add a 3DNS to a SyncGroup screen opens.
4. In the list of 3DNS Controllers, check the box next to the IP address of each 3DNS Controller that you want to add to the sync group. Select the principal 3DNS Controller first.
5. Click **Update**.

To define a sync group manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Use the syntax shown in Figure 4.16 to define sync groups.

The **sync_group** statement should appear after the **datacenter** statement and before **server** statements.

```
sync_group {  
    name "<name>"  
    3dns <ip_address | "domain_name">  
    [ 3dns <ip_address | "domain_name"> ] ...  
}
```

Figure 4.16 Syntax for setting up a sync group

Figure 4.17 shows a sample **sync_group** statement:

```
sync_group {  
    name "sync"  
    3dns 192.168.101.2    // New York  
    3dns 192.168.102.2    // Los Angeles  
}
```

Figure 4.17 Syntax for setting up a sync group

Setting the time tolerance value

The time tolerance value is a global variable that defines the number of seconds that one 3DNS Controller's time setting is allowed to be out of sync with another 3DNS Controller's time setting. See *Understanding how the time tolerance variable affects sync groups*, on page 2-9 for details.

To check the value for the time tolerance setting using the F5 Configuration utility

1. In the navigation pane, click **System**.
The System - General screen opens.
2. On the toolbar, click **Timers & Task Intervals**.
3. Note the value in the **3DNS Sync Time Tolerance** box, and change it if necessary.
4. If you change this setting, click **Update** to save it.

To check the value for the time tolerance setting in the configuration file

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Search for **time_tolerance**. If you don't find the **time_tolerance** sub-statement in the configuration file, the default (**10**) is used.

Configuring global variables

Default values for global parameters are adequate for most situations. However, we do recommend that you specifically enable encryption for US 3DNS Controllers.

To configure global parameters using the F5 Configuration utility

1. In the navigation pane, click **System**.
The System - General screen opens. Note that global parameters are grouped into several categories on this screen. Each category has its own toolbar item, and online help is available for each parameter.
2. Make general global changes at the System - General screen or, to make changes to global parameters in other categories, click the appropriate toolbar item.
3. Enter the value you want to change in the appropriate box.

For example, to enable encryption for iQuery transactions (which is recommended), go to the System - General screen and check the **Enabled** box. If you want to use a non-default name for the encryption key file, type it in the **Key File** box.

4. Click **Update**.

The new global parameters are added to your configuration.

To configure global parameters manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Locate or add the **globals** statement. The **globals** statement should be at the top of the file.
3. Under the **globals** statement, type the appropriate sub-statement and value.

For example, to enable encryption for iQuery transactions (which is recommended), change the encryption parameter to **yes** (the default setting is **no**). If you want to use a non-default name for the encryption key file, type it on the next line.

Figure 4.18 shows the correct syntax for enabling encryption:


```
globals {  
    encryption yes  
    encryption_key_file "/etc/F5key.dat"  
}
```

Figure 4.18 Syntax for enabling encryption

For descriptions of all global parameters, see *The globals statement*, on page A-6.

Configuring IP filters

Filters control network traffic by specifying whether packets are accepted or rejected by the 3DNS Controller. Filters apply to both incoming and outgoing traffic. When creating a filter, you define the criteria to apply to each packet that is processed by the 3DNS Controller. You can configure the 3DNS Controller to accept or block each packet based on whether the packet matches the criteria.

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.

Defining the filter criteria

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.

To define an IP filter using the F5 Configuration utility

1. Click **IP Filters** in the navigation pane.
The IP Filters screen opens.
2. In the IP Filters screen, click **Add Filter**.
The Add IP Filter screen opens.
3. 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. If you want to filter traffic based on its source, specify the client source information:
  - a) In the **Source IP Address** box, enter the IP address from which you want to filter traffic.
  - b) In the **Source Port** box, enter the port number from which you want to filter traffic.
3. If you want to filter traffic based on its destination, specify the destination information:
  - a) In the **Destination IP Address** box, enter the IP address to which you want to filter traffic.
  - b) In the **Destination Port** box, enter the port number to which you want to filter traffic.
3. Click **Add** to add the IP filter to the system.

◆ Note

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

Configuring Sendmail

You can configure the 3DNS Controller to send email notifications to you, or to other administrators. The 3DNS Controller includes a sample Sendmail configuration file that you can use to start with, but you will have to customize the Sendmail setup for your network environment before you can use it.

Before you begin setting up Sendmail, you may need to look up the name of the mail exchanger for your domain. If you already know the name of the mail exchanger, go to *Setting up Sendmail*, on page 4-34 for details about setting up the **sendmail** daemon itself.

Finding the mail exchanger for your domain

You can use the **nslookup** command on any workstation that is configured for lookup. Once you find the primary IP address for your domain, you can find the mail exchanger for your domain.

To find the mail exchanger

1. Identify the default server name for your domain. From a workstation capable of name resolution, type the following on the command line:

```
/etc# nslookup
```

The command returns a default server name and corresponding IP address:

```
Default Server: <server name>  
Address: <server>
```

2. Use the domain name to query for the mail exchanger:

```
set q=mx  
<domain name>
```

The returned information includes the name of the mail exchanger. For example, the sample information shown in Figure 4.19 lists **bigip.net** as the preferred mail exchanger.

```

bigip.net    preference = 10, mail exchanger = mail.SiteOne.com
bigip.net    nameserver = ns1.bigip.net
bigip.net    nameserver = ns2.bigip.net
bigip.net    internet address = 192.17.112.1
ns1.bigip.net      internet address = 192.17.112.2
ns2.bigip.net      internet address = 192.17.112.3

```

Figure 4.19 Sample mail exchanger information

Setting up Sendmail

When you actually set up Sendmail, you need to open and edit a couple of configuration files. Note that the 3DNS Controller does not accept email messages, and that you can use the **crontab** utility to purge unsent or returned messages, and that you can send those messages to yourself or another administrator.

To set up and start Sendmail

1. Copy **/etc/sendmail.cf.off** to **/etc/sendmail.cf**.
2. To set the name of your mail exchange server, open the **/etc/sendmail.cf** file and set the DS variable to the name of your mail exchanger. The syntax for this entry is:

DS<MAILHUB_OR_RELAY>

3. Save and close the **/etc/sendmail.cf** file.
4. To allow Sendmail to flush outgoing messages from the queue for mail that cannot be delivered immediately, open the **/etc/crontab** file, and change the last line of the file to read:

```
0,15,30,45 * * * * root /usr/sbin/sendmail -q > /dev/null 2>&1
```

5. Save and close the **/etc/crontab** file.
6. To prevent returned or undelivered email from going unnoticed, open the **/etc/aliases** file and create an entry for **root** to point to you or another administrator at your site.

root: networkadmin@SiteOne.com

7. Save and close the **/etc/aliases** file.
8. Run the **newaliases** command to generate a new aliases database that incorporates the information you added to the **/etc/aliases** file.
9. To turn Sendmail on, either reboot the system or type the following command:

/usr/sbin/sendmail -bd -q30m



5

Configuring Basic Load Balancing

- **Getting started with load balancing**
- **Defining a wide IP**
- **Changing global variables that affect load balancing**

Getting started with load balancing

The third and final phase of installing 3DNS Controllers is to configure load balancing. The 3DNS Controller uses the load balancing modes you configure to determine how to handle DNS name resolution requests. Name resolution requests are sent from a local DNS and are ultimately resolved by a virtual server. The virtual server is managed by a BIG/ip Controller or host, and the BIG/ip Controller or host is managed by the 3DNS Controller.

This chapter describes how to configure load balancing. Follow the procedures in this chapter to configure any load balancing mode. If you want to implement more advanced load balancing modes, see Chapter 6, *Configuring Specialized Load Balancing*.

You can configure load balancing at the wide IP and global levels:

❖ **Wide IP**

Most load balancing configuration takes place at the wide IP level, which means that each pool of virtual servers, contained in a wide IP definition, uses the specified load balancing mode. You define which load balancing modes to use when you define the wide IP. See *Defining a wide IP*, in the following section. You can actually set up to three load balancing modes for each pool: a preferred mode, and two modes to use in case the preferred mode fails.

❖ **Global**

At the global level, you can set default alternate and default fallback modes. If you don't configure alternate or fallback modes within a wide IP definition, the 3DNS Controller uses the global alternate and fallback modes. Configuring global alternate and fallback modes is described on page 5 -10.

Defining a wide IP

After you configure BIG/ip Controllers, hosts, and the virtual servers they manage, you need to group the configured virtual servers into a wide IP. A wide IP is a mapping of a fully qualified domain name to a set of virtual servers that host the domain content, such as a web site or an e-commerce site.

Before defining the first wide IP, you should do the following:

- ❖ Gather your BIG/ip Controller and host configuration information so that you can easily see which virtual servers have the replicated content. Then you can decide how to group virtual servers into pools.
- ❖ Decide which load balancing modes to use for each pool of virtual servers. If you need to review the available load balancing modes, see *Choosing a load balancing mode*, on page 2-26.

◆ Note

*NameSurfer, an application included with the 3DNS Controller, sets up DNS zone files so that wide IP definitions are properly linked to DNS. No action is required on your part, as NameSurfer automatically handles this process. For more information on NameSurfer, see the online help that is included with it (available from the F5 Configuration utility). If you do not want to configure the 3DNS Controller manually, see **Relating BIND information to 3DNS Controller wide IP definitions**, on page C -10.*

There may be situations (for example, for e-commerce and ftp sites) where you need to configure a wide IP so that connections are not sent to a given address unless multiple ports or services are available. You configure this behavior after the wide IP is defined. For details, see *Setting up load balancing for services that require multiple ports*, on page 6-10.

Understanding pools

A wide IP contains one or more pool definitions. A pool is a group of virtual servers that the 3DNS Controller load balances. You can include both types (BIG/ip Controller and host) of virtual servers in a pool definition.

◆ Note

*At this time the F5 Configuration utility does not support using more than two pools for load balancing. Also, you cannot configure the names of pools using the Configuration utility. If your **wideip.conf** file includes multiple pools, do not use the F5 Configuration utility.*

Adding a wide IP

After determine which virtual servers belong in which wide IP pools, you are ready to add the first wide IP. If you want to configure a wide IP for a domain that has one registered alias name, you can use the Configuration utility as described below. To configure wide IPs that have multiple aliases, you must configure the wide IP manually, as described on page 5 -5.

To define a wide IP using the F5 Configuration utility

1. In the navigation pane, click **Wide IPs**.
The Wide IP List screen opens.
2. On the toolbar, click **Add Wide IP**.
3. In the **Wide IP Name** box, type a name for the wide IP.
4. In the **Wide IP Alias** box, type a domain name alias only if the domain has a registered alias name.
5. In the **Wide IP Address** box, type the IP address of the wide IP.
6. In the **Wide IP Port** box, type a port number or select a service from the list.

7. If you want to use a TTL other than the default (as specified on the System - Timers & Task Intervals screen), type a new TTL value in the **Wide IP TTL** box. Specifying a new value here overrides the global default value only for this wide IP. It overrides the TTL in the resource record.
8. In the **Select Virtual Server Type** box, check one of the following: **VSb** if the pool contains virtual servers that run on BIG/ip Controllers, or **VSh** if the pool contains virtual servers that run on hosts.
9. Click **Next**.
The Configure Load Balancing screen opens.
10. If you are setting up two pools that use Ratio mode, type the ratio weight for the pool in the **Pool Ratio** box.
11. In the **Preferred** box, select the load balancing mode to use for this wide IP. The default is **Round Robin**.
12. If you want to use an alternate method other than the default (as specified on the System - Load Balancing screen), select the load balancing mode to use if the Preferred mode fails in the **Alternate** box. The default is **Round Robin**.
13. If you want to use a fallback method other than the default (as specified on the System - Load Balancing screen), select the load balancing mode to use if the Preferred and Alternate modes fail. The default is **Return to DNS**.
14. If you want to use the Round Robin LDNS feature, check **Use Round Robin LDNS**. When you turn on Round Robin LDNS, the 3DNS Controller returns a list of available virtual servers available for load balancing to a client and stores the list in the browser cache. If you check this option, also enter the maximum number of virtual servers that the 3DNS Controller should return in the **Round Robin LDNS Limit** box. The default, **0**, specifies that the 3DNS Controller returns the IP addresses of all (up to 16) available virtual servers.

15. If you want to use the Dynamic Ratio option, check **Use Dynamic Ratio**. This option is relevant only if you are using the Round Trip Time, Completion Rate, Hops, or Quality of Service mode. When you turn on Dynamic Ratio, the 3DNS Controller treats Quality of Service (QOS) scores as ratios, and it uses each server in proportion to the ratio determined by the QOS calculation. For more information, see *Using the Dynamic Ratio option*, on page 6-6.
16. If you selected Quality of Service (QOS) as the Preferred or Fallback mode, you can use the **Quality of Service** group to fine-tune the QOS coefficients. Before doing so, we recommend that you read *Understanding QOS coefficients*, on page 6-2.
17. Click **Next**.
The Select Virtual Servers screen opens.
18. From the list, select the virtual servers you want to add to this wide IP's pool.
19. If you selected ratio as a load balancing mode, assign a ratio value to each virtual server in the **Ratio** column.
20. If you selected global availability as a load balancing mode, rank the virtual servers in descending order of preference in the **Order** column.

Global availability is a specialized mode. For more information, see *Setting up Global Availability mode*, on page 6-8.

21. Click **Finish**.

The wide IP is added to your configuration. Repeat this process for each wide IP you want to add.

To define a wide IP manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Add a **wideip** statement.

Place the **wideip** statement after all **server** statements and before any **topology** statement.

3. Under the **wideip** statement, enter the wide IP's address, port, and name information. Enclose the wide IP's name in quotation marks.
4. Configure any options you want to set (such as the TTL, port list, or QOS coefficients) by entering the appropriate sub-statements.
5. Define the **pool** sub-statement. At the minimum, the **pool** sub-statement should include its name (enclosed in quotation marks) and the virtual servers it contains.
6. Define the load balancing modes you want to use by entering **preferred**, **alternate**, and **fallback** sub-statements.
7. Define the IP address, port, and ratio value for each virtual server that you want to include in this pool.

Figure 5.1 shows the correct syntax for the **wideip** statement:

```

wideip {
  address <ip_addr>
  port <port_number> | <"service name">
  name <"domain_name">
  [ alias <"alias_name"> ]
  [ ttl <number> ]
  [ port_list <port_number> <port_number> ... ]
  [ qos_coeff {
    rtt <n>
    completion_rate <n>
    packet_rate <n>
    topology <n>
    hops <n>
  } ]
  [ pool_lbmode <rr | ratio | ga | random> ]
  pool {
    name <"pool_name">
    [ ratio <pool_ratio> ]
    [ dynamic_ratio < yes | no > ]
    [ rr_ldns < yes | no > ]
    [ preferred < completion_rate | ga | hops | leastconn |
      packet_rate | qos | random | ratio | return_to_dns | rr |
      rtt | topology > ]
    [ alternate < ga | null | random | ratio | return_to_dns |
      rr | topology > ]
    [ fallback < completion_rate | ga | hops | leastconn | null |
      packet_rate | qos | random | ratio | return_to_dns | rr |
      rtt | topology > ]
    address <vs_addr>[:<port>] [ratio <weight>]
  }
}

```

Figure 5.1 Syntax for the wideip statement

Figure 5.2 shows a sample **wideip** statement. This statement defines a wide IP named **mx.wip.domain.com**, with an alias of **mail.wip.domain.com**. The wide IP contains two pools, with **pool_1** receiving three times as many requests as **pool_2**. The 3DNS Controller attempts to resolve requests sent to **pool_1** using the Round Trip Times (RTT) method. This method sends

connections to the virtual server in the pool that demonstrates the best round trip time between the virtual server and the client LDNS. If the 3DNS Controller cannot resolve the request using the RTT mode, the controller distributes requests randomly. The 3DNS Controller distributes requests to **pool_2** to the two defined virtual servers, at a 2:1 ratio where the first listed virtual server receives twice as many connections as the second.

```
wideip {
  address      192.168.102.50
  service      "smtp"
  name         "mx.wip.domain.com"
  alias        "mail.wip.domain.com"
  pool_lbmode   ratio
  pool {
    name        "pool_1"
    ratio       3
    preferred    rtt
    alternate    random
    address     192.168.101.50
    address     192.168.102.50
    address     192.168.103.50
  }
  pool {
    name        "pool_2"
    ratio       1
    preferred    ratio
    address     192.168.104.50    ratio 2
    address     192.168.105.50    ratio 1
  }
}
```

Figure 5.2 Example syntax for defining a wide IP

Troubleshooting manual configuration problems

Adding a wide IP is a process that requires careful planning and use of correct syntax. The following recommendations are intended to make it easier for you to spot and resolve any configuration problems:

❖ F5 Configuration utility

The F5 Configuration utility contains a Statistics area that is useful in diagnosing problems, as it provides a snapshot of your 3DNS Controller network at any given time. To use it, click **Statistics** in the left pane of the F5 Configuration utility, then click either **Summary** (and scroll until you see the **Wide IP** table) or **Wide IPs**.

❖ wideip.conf syntax

If you manually configure wide IPs, use the **3dparse** utility to verify **wideip.conf** syntax before you start **named**. To use this utility, type **3dparse** on the command line. For details on the **3dparse** utility, see page B -1. For an example of a **wideip.conf** file, see Appendix A, *Wideip.conf Syntax*.

❖ /var/log/messages

If you encounter an error that you cannot trace, open the **/var/log/messages** file on your system. Using the UNIX **grep** utility, search for "named" (for example, **tail -100 /var/log/messages | grep named**). This log file saves verbose error information, and should contain an explanation of the error.

❖ BIND syntax

If you are setting up the configuration manually, you may want to refer to one of the following BIND resources for help and background information:

- Appendix C of this manual
- The O'Reilly & Associates book, *DNS and BIND*, 3rd Edition
- <http://www.isc.org/bind.html>

Changing global variables that affect load balancing

You can configure global variables that affect how load balancing is handled on a global basis, for all wide IPs. You can override these global settings for individual wide IPs as necessary.

Global variables that affect load balancing fall into two categories:

- ❖ Alternate and fallback load balancing modes
- ❖ TTL and timer values

The default settings for these variables are adequate for most configurations. However, if you want to change any global variable, you should refer to the online help or to *The globals statement*, on page A-6.

Setting global alternate and fallback modes

You can decide which load balancing modes to use for all wide IPs in the event that a wide IP's preferred mode fails.

To configure global alternate and fallback load balancing modes using the F5 Configuration utility

1. In the navigation pane, click **System**.
The System - General screen opens.
2. On the toolbar, click **Load Balancing**.
3. In the **Default Alternate** box, specify the load balancing mode to use should a wide IP's preferred mode fails.
4. In the **Default Fallback** box, specify the load balancing mode to use should the preferred and alternate modes fail.

If all modes fail, requests are returned to DNS.

To configure global alternate and fallback load balancing modes manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Locate or add the **globals** statement. The **globals** statement should be at the top of the file.
3. Use the syntax shown in Figure 5.3 to define global alternate and fallback load balancing modes.

```
globals {
  [ default_alternate < ga | null | random | ratio |
    return_to_dns | rr | topology > ]
  [ default_fallback < completion_rate | ga | hops | leastconn |
    null | packet_rate | qos | random | ratio | return_to_dns |
    rr | rtt | topology > ]
}
```

Figure 5.3 Configuring global alternate and fallback load balancing modes

Figure 5.4 shows a sample **globals** statement that defines global load balancing variables:

```
globals {
  default_alternate leastconn
  default_fallback rr
}
```

Figure 5.4 Sample syntax for setting global load balancing variables

Understanding TTL and timer values

Each 3DNS object has an associated time-to-live (TTL) value. A TTL is the amount of time (measured in seconds) for which metrics information is considered valid. After a timeout is reached, the 3DNS Controller refreshes the information.

Table 5.1 describes each TTL value, as well as its default setting.

Parameter	Description	Default
BIG/ip TTL	Specifies the amount of time (in seconds) that BIG/ip Controller information is to be used by the 3DNS Controller for name resolution and load balancing.	60
Host TTL	Specifies the amount of time (in seconds) that generic host machine information is to be used by the 3DNS Controller for name resolution and load balancing.	240
3DNS TTL	Specifies the amount of time (in seconds) that the 3DNS Controller considers performance data for the other 3DNS Controllers to be valid.	60
Virtual server TTL	Specifies the amount of time (in seconds) that virtual server information (data acquired from a BIG/ip Controller or other host machine about a virtual server) is to be used by the 3DNS Controller for name resolution and load balancing.	120
Trace TTL	Specifies the amount of time (in seconds) that the 3DNS Controller considers trace route data to be valid.	604800 (seven days)
Path TTL	Specifies the amount of time (in seconds) that path information is to be used by the 3DNS Controller for name resolution and load balancing.	600
Default TTL	Specifies the default A record TTL value for wide IPs. If you do not specify a TTL value in a wide IP definition, it takes on the value of default_ttl .	30

Table 5.1 TTL values

Each 3DNS object also has a timer value. A timer value defines the frequency (measured in seconds) at which the 3DNS Controller refreshes the metrics information it collects. In most cases, the default values for the TTL and timer parameters are adequate. However, if you make changes to any TTL or timer value, keep in mind that an object's TTL value must be greater than its timer value.

Table 5.2 describes each timer value, as well as its default setting.

Parameter	Description	Default
BIG/ip data refresh	Specifies how often the 3DNS Controller refreshes BIG/ip Controller information.	20
Host data refresh	Specifies how often the 3DNS Controller refreshes other host machine information.	90
3DNS data refresh	Specifies the frequency (in seconds) for retrieving performance data for other 3DNS Controllers in the sync group.	20
Virtual server data refresh	Specifies how often the 3DNS Controller refreshes virtual server information.	30
Trace data refresh	Specifies the frequency (in seconds) for retrieving trace route data (trace routes between each data center and each local DNS).	60
Path data refresh	Specifies how often the 3DNS Controller refreshes path information (for example, round trip time or ping packet completion rate).	120
Remote nodes query	Specifies how often the 3DNS Controller queries remote 3DNS Controllers and BIG/ip Controllers.	60

Table 5.2 *Time values*

To configure global TTL and timer values using the F5 Configuration utility

1. In the navigation pane, click **System**.
The System - General screen opens.
2. If you want to configure the default TTL for wide IPs, enter a new value in the **Default TTL** box.
3. To configure other TTL and timer values, click **Timers & Task Intervals** on the toolbar.
The System - Timers & Task Intervals screen opens.
4. Make any edits you need to make in the appropriate boxes, and click **Update**.

To configure global TTL and timer values manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Locate or add the **globals** statement. The **globals** statement should be at the top of the file.
3. Use the syntax shown in Figure 5.5 to define global TTL and timer values.

```
globals {  
  [ timer_get_3dns_data <number> ]  
  [ timer_get_bigip_data <number> ]  
  [ timer_get_host_data <number> ]  
  [ timer_get_vs_data <number> ]  
  [ timer_get_path_data <number> ]  
  [ timer_get_trace_data <number> ]  
  [ timer_check_keep_alive <number> ]  
  [ 3dns_ttl <number> ]  
  [ bigip_ttl <number> ]  
  [ host_ttl <number> ]  
  [ vs_ttl <number> ]  
  [ path_ttl <number> ]  
  [ trace_ttl <number> ]  
  [ default_ttl <number> ]  
}
```

Figure 5.5 Syntax for configuring global TTL and timer values



6

Configuring Specialized Load Balancing

- **Configuring specialized load balancing using advanced modes**
- **Setting up Quality of Service (QOS) mode**
- **Setting up Global Availability mode**
- **Setting up load balancing for services that require multiple ports**
- **Setting up topology-based features**

Configuring specialized load balancing using advanced modes

This section describes the following specialized, or more advanced, load balancing modes:

- ❖ Quality of service
- ❖ Global availability
- ❖ E-commerce
- ❖ Topology access control
- ❖ Topology load balancing

You configure a specialized load balancing mode in the same way that you configure the more basic modes: by editing the wide IP definition. See *Defining a wide IP*, on page 5-2. You can also set global alternate and fallback load balancing modes. See *Setting global alternate and fallback modes*, on page 5-10.

Setting up Quality of Service (QOS) mode

The **Quality of Service (QOS)** mode is a user-definable mode that includes a configurable combination of the RTT, Completion Rate, Packet Rate, Topology, and Hops modes. The QOS mode is based on an equation that takes each of these performance factors into account. The default setting is for each factor to be weighted equally, and you can easily use the Quality of Service mode with these default settings simply by specifying it as your preferred load balancing mode.

If you want to change the settings, you can customize the equation to put more or less weight on each individual factor. When the 3DNS Controller selects a virtual server, it chooses the server with the best overall score.

Understanding QOS coefficients

The following table lists each QOS coefficient, its scale, a likely upper limit for each, and whether a higher or lower value is more efficient.

Coefficient	How measured	Example upper limit	Higher or lower?
Packet rate	Packets per second	700	Lower
Round trip time	Microseconds	2,000,000	Lower
Completion rate	Percentage of successfully transferred packets (0-100%)	100%	Higher
Topology	Score that defines network proximity by comparing server and local DNS IP addresses (0-2 ³²)	100	Higher
Hops	Number of hops	64	Lower

Table 6.1 QOS coefficients: ranges and limits

If you want to change the default QOS coefficients, keep the following issues in mind.

❖ **Scale**

The raw metrics for each coefficient are not on the same scale. For example, completion rate is measured in percentages while the packet rate is measured in packets per second.

❖ **Normalization**

The 3DNS Controller normalizes the raw metrics to values in the range of 0 to 10. As the QOS value is calculated, a high measurement for completion rate is good, because a high percentage of completed connections are being made, but a high value for packet rate is not desirable because you are trying to find a virtual server that is not overly taxed at the moment.

❖ Emphasis

You can adjust coefficients to emphasize one normalized metric over another. For example, by changing the coefficients to the values shown below, you are putting the most emphasis on completion rate.

Example: Emphasizing completion rate

```
globals {  
    qos_coeff_rtt 20  
    qos_coeff_completion_rate 100  
    qos_coeff_packet_rate 50  
    qos_coeff_topology 0  
    qos_coeff_hops 0  
}
```

Figure 6.1 *Emphasizing completion rate*

In the above example, if completion rates for two virtual servers are close, the virtual server with the best packet rate is chosen. If both completion rates and packet rates are close, the round trip time (rtt) breaks the tie. In this example, the topology and hops modes are not used in determining how to distribute connections.

Customizing the QOS equation

You can customize the QOS equation globally, meaning that the equation applies to all wide IPs that use the QOS mode. You can also override the global QOS equation settings for individual wide IPs as necessary.

You must specify QOS as a load balancing mode before you can customize its equation.

To assign global QOS coefficients using the F5 Configuration utility

1. In the navigation pane, click **System**.
The System - General screen opens.
2. On the toolbar, click **Load Balancing**.
The System - Load Balancing screen opens.
3. Define the global QOS coefficients in the **Round Trip Time, Completion Rate, Hops, Packet Rate**, and **Topology** boxes.
4. Click **Update** for the changes to take effect.

To assign QOS coefficients for a specific wide IP using the F5 Configuration utility

1. In the navigation pane, click **Wide IPs**.
2. Click a wide IP name.
The Edit Wide IP screen opens.
3. On the toolbar, click **Modify VSb Pool** or **Modify VSh Pool**, depending on which type of pool you have.
4. Define the wide IP's QOS coefficients in the **Round Trip Time, Completion Rate, Hops, Packet Rate**, and **Topology** boxes.
5. Click **Update** for the changes to take effect.

To assign global QOS coefficients manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Locate or add the **globals** statement. The **globals** statement should be at the top of the file.
3. Refer to the example syntax shown in Figure 6.2 to define a global QOS equation.

```
globals {  
    qos_coeff_rtt 20  
    qos_coeff_completion_rate 5  
    qos_coeff_packet_rate 3  
    qos_coeff_topology 0  
    qos_coeff_hops 0  
}
```

Figure 6.2 Sample global QOS equation

To assign QOS coefficients for a specific wide IP manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Locate the **wideip** statement you want to edit.
3. Refer to the example syntax shown in Figure 6.3 to define a wide IP's QOS equation. Figure 6.3 displays a wide IP definition that uses overrides for the global settings shown in Figure 6.2.

```

wideip {
  address          192.168.101.50
  service          "http"
  name             "www.wip.domain.com"
  ttl              60      // increase the domain default ttl
  qos_coeff {
    rtt            21
    hops           0
    completion_rate 7
    packet_rate    5
    topology       1
  }
  pool {
    name           "Pool_1"
    ratio          2      // applies to pool_lbmode == ratio
    preferred      qos
    alternate      ratio
    address        192.168.101.50    ratio 2
    address        192.168.102.50    ratio 1
    address        192.168.103.50    ratio 1
  }
  pool {
    name           "Pool_2"
    ratio          1
    preferred      rr
    address        192.168.102.60    ratio 2
    address        192.168.103.60    ratio 1
  }
}

```

Figure 6.3 QOS coefficient settings that override the global default settings

Using the Dynamic Ratio option

When the Dynamic Ratio option is turned on, the 3DNS Controller treats QOS scores as ratios, and it uses each server in proportion to the ratio determined by the QOS calculation. When the Dynamic Ratio option is turned off (the default), the 3DNS Controller uses only the server with the highest QOS score for load balancing ("winner takes all") until metrics information is refreshed.

To turn on the Dynamic Ratio option using the F5 Configuration utility

1. In the navigation pane, click **Wide IPs**.
2. Click a wide IP name.
The Edit Wide IP screen opens.
3. On the toolbar, click **Modify VSb Pool** or **Modify VSh Pool**, depending on which type of pool you have.
4. Check **Use Dynamic Ratio**.
5. Click **Update** for the change to take effect.

To turn on the Dynamic Ratio option manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Locate the **wideip** statement and the pool definition you want to edit.
3. Add the syntax (shown in bold) to the pool definition.

```
pool {
    name <"pool_name">
    [ ratio <pool_ratio> ]
    dynamic_ratio yes
    [ rr_ldns < yes | no > ]
    [ rr_ldns_limit <number> ]
    [ preferred < completion_rate | ga | hops | leastconn |
      packet_rate | qos | random | ratio | return_to_dns | rr |
      rtt | topology > ]
    [ alternate < ga | null | random | ratio | return_to_dns |
      rr | topology > ]
    [ fallback < completion_rate | ga | hops | leastconn | null |
      packet_rate | qos | random | ratio | return_to_dns | rr |
      rtt | topology > ]
    address <vs_addr>[:<port>] [ratio <weight>]
}
```

Figure 6.4 Enabling dynamic ratio

Setting up Global Availability mode

Figure 6.5 shows the 3DNS Controller using the global availability load balancing mode. The global availability mode selects the first available virtual server in a wide IP definition. If that virtual server becomes unavailable, subsequent connections go to the next listed virtual server in the wide IP definition.

The 3DNS Controller always starts with the first virtual server in the list. Over time, the first server in the list receives the most connections, and the last server in the list receives the fewest connections.

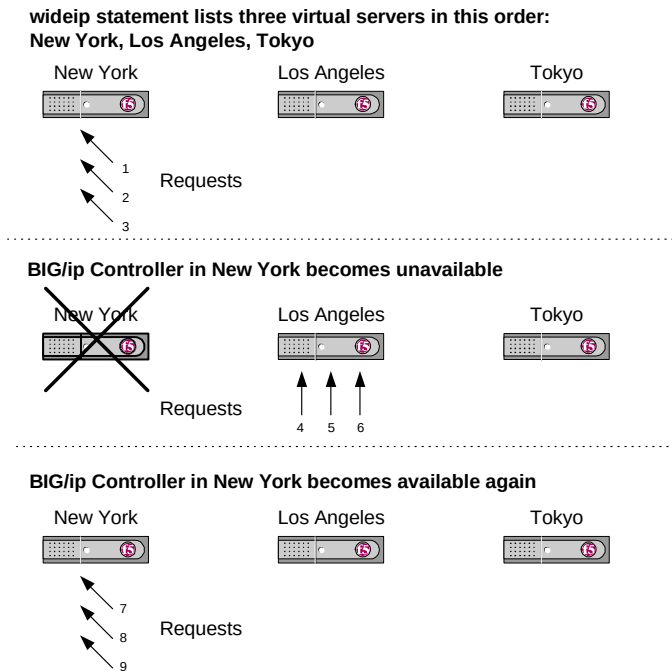


Figure 6.5 Global Availability mode

To implement the global availability load balancing mode using the F5 Configuration utility

1. In the navigation pane, click **Wide IPs**.
2. Click a wide IP name.
The Edit Wide IP screen opens.
3. On the toolbar, click **Modify VSb Pool** or **Modify VSh Pool**, depending on which type of pool you have.
4. Select global availability as the preferred, alternate, or fallback load balancing mode.
5. Click **Next**.
The Select Virtual Servers screen appears.
6. From the list, select the virtual servers you want to add to this wide IP's pool, as well as the order in which connections should be sent to them.
7. Click **Update** for the changes to take effect.

To implement the global availability load balancing mode manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Locate the **wideip** statement you want to edit.
3. Define global availability as the preferred, alternate, or fallback load balancing mode.
4. List the virtual servers in descending order of preference.
See the example, next.

A Global Availability configuration example

With the global availability load balancing mode, you can configure one data center as your primary service and have several alternate services on standby. In the **wideip** statement, list the virtual servers in descending order of preference. The first available virtual server is chosen for each resolution request.

Figure 6.6 shows a sample **wideip** definition where global availability is the preferred load balancing mode.

```
// Global availability
wideip {
    address      192.168.101.60
    port         80 // http
    name         "cgi.wip.domain.com"
    pool {
        name     "mypool"
        preferred ga
        address   192.168.101.60
        address   192.168.102.60
        address   192.168.103.60
    }
}
```

Figure 6.6 *Configuring a standby data center*

All resolution requests are sent to the first listed virtual server (**192.168.101.60** in this example) unless the first listed virtual server becomes unavailable for any reason. If the first listed virtual server becomes unavailable, resolution requests are sent to the second listed virtual server, and so on.

Setting up load balancing for services that require multiple ports

Some sites require that you use multiple ports or services to access them. For these cases, you can configure a wide IP so that connections are not sent to a given address unless all specified ports or services are available.

To configure multiple ports for a wide IP using the F5 Configuration utility

1. In the navigation pane, click **Wide IPs**.

2. Click a wide IP name.
The Edit Wide IP screen opens.
3. On the toolbar, click **Port List**.
The Wide IP Port List screen opens.
4. Type a port number in the box or select a service from the list, then click the right arrow button.
5. Repeat step 4 for each port or service you need to add.
6. Click **Update**.

To configure multiple ports for a wide IP manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Locate the **wideip** statement you want to edit.
3. Add the **port_list** line as indicated in bold in Figure 6.7.

```
wideip {
    address <ip_addr>
    port <port_number> | <"service name">
    name <"domain_name">
    [ alias <"alias_name"> ]
    [ ttl <number> ]
    [ port_list <port_number> <port_number> ... ]
    [ qos_coeff {
        rtt <n>
        completion_rate <n>
        packet_rate <n>
        topology <n>
        hops <n>
    } ]
    [ pool_lbmode <rr | ratio | ga | random> ]
    [ pool definitions ...]
```

Figure 6.7 Enabling the *port_list* option

An example configuration for e-commerce services

In this example, you are setting up a site for selling a product on the Internet. This site contains secure and non-secure areas. The non-secure area contains the product catalog and the secure area is for placing orders. You can configure a wide IP so that clients are only sent to a virtual server if both the secure and non-secure areas are available.

The key entry here is **port_list**. The **port_list** entry specifies that requests can only be sent to virtual servers in this pool if ports 80 (non-secure area) and 443 (secure area) are available.

```
wideip {
  address      192.168.101.70
  port         80 // http
  port_list    80 443          // e-commerce
  name         "ssl.wip.domain.com"
  pool_lbmode  rr
  pool {
    name       "bigip_pool"
    ratio      2
    preferred   qos
    alternate   ratio
    address    192.168.101.70  ratio 7
    address    192.168.102.60  ratio 2
  }
  pool {
    name       "host_pool"
    ratio      1
    preferred   ratio
    address    192.168.104.50  ratio 2
    address    192.168.105.60  ratio 1
  }
}
```

Figure 6.8 Syntax for e-commerce services

For each virtual server address in the pool, a virtual server must exist for each port in the port list. In the above example, the BIG/ip Controllers and host machines must have the following virtual servers defined:

```
192.168.101.70:80
192.168.101.70:443
192.168.102.60:80
192.168.102.60:443
192.168.104.50:80
192.168.104.50:443
192.168.105.60:80
192.168.105.60:443
```

Setting up topology-based features

The 3DNS Controller supports two topology-based features: topology-based access control, and a topology load balancing mode.

To use the topology load balancing mode, you should first set up topology-based access control so that the list records are defined. Otherwise, the topology load balancing mode has no effect.

Setting up topology-based access control is described next. Using topology as a load balancing mode is described on page 6-22.

Setting up Topology-based access control

You can use topology-based access control to implement a form of wide-area IP filtering. Topology-based access control allows you to specify which data centers are acceptable for a given resolution request, based on the proximity of the data center's IP address to the requesting local DNS server's IP address.

Understanding the list records

The **list records** in the topology definition define a score for pairs of known local DNS servers and data centers.

For example, here is a sample list record:

```
192.168.101.0/24    198.0.0.0/24    6
```

Essentially, each record defines two network endpoints in CIDR (Classless Interdomain Routing) format, and a score. The CIDR format consists of an IP address and a number n designating a subnet bitmask. The bitmask is made up of n ones followed by $32 - n$ zeros. For example, for $n = 8$, the bitmask is:

```
11111111000000000000000000000000
\_____/ \_____/
      8 ones      24 zeros
```

The first endpoint, **A**, corresponds to the IP address of a server (either a BIG/ip Controller or a host). The second endpoint, **B**, corresponds to the IP address of the local DNS. Suppose a local DNS, **L**, requests a name resolution from the 3DNS Controller, and the virtual server being considered as an answer is managed by a BIG/ip Controller, **S**. The list record that matches is the one where the following equation is TRUE:

```
((S & A-mask == A & A-mask) && (L & B-mask == B & B-mask))
```

Referring to the example **topology** statement that starts on page 6 - 19, say that the local DNS **198.0.0.0** requested name resolution for **www.domain.com**, and a virtual server in the pool belonged to the BIG/ip Controller **192.168.101.0**. In this scenario, the 3DNS Controller considers the first list record to be a match.

Note that in the above list record, the single ampersand (&) is a bitwise operator, and the double ampersands (&&) are logical operators.

Understanding the topology score

Each list record includes a score, which is used both in topology-based load balancing, and in topology-based access control. If multiple list records in a **topology** statement have the exact same server IP/mask and local DNS IP/mask but have different scores, only the last record is declared valid. For example, the first set of records is equivalent to the second set of records.

192.168.101.0/24	198.0.0.0/24	6	
192.168.101.0/8	198.0.0.0/8	1	
192.168.101.0/24	198.0.0.0/24	89	<-- replaces 1st record
192.168.101.0/24	198.0.0.0/24	0	<-- replaces previous record
192.168.101.0/24	198.0.0.0/24	3	<-- replaces previous record

This set of records is equivalent to the above set of five records.

192.168.101.0/8	198.0.0.0/8	1
192.168.101.0/24	198.0.0.0/24	3

Using the longest match rule

The 3DNS Controller uses the same type of longest match rule that routers commonly use. If there are several IP/mask items that match a particular IP address, the 3DNS Controller selects the record that is most specific, and thus has the longest mask (n is the largest).

For example, 192.168.101.4 matches 192.168.101.4/0, 192.168.101.4/8, 192.168.101.4/13, 192.168.101.4/24, and 192.168.101.4/32, but the longest matching IP/mask is 192.168.101.4/32. When the **longest_match** parameter is set to **yes** (the default), the longest match rule is obeyed for local DNS IP addresses, and also for server IP addresses, when there are multiple matches for a server/local DNS combination. This means that for the virtual server **192.168.101.50** owned by BIG/ip Controller **192.168.101.40** and local DNS **198.0.0.40**, the third list record is the longest match:

192.168.101.0/24	198.0.0.40/24	2
192.168.101.0/8	198.0.0.40/16	0
192.168.101.0/8	198.0.0.40/27	6 <-- Longest Match
192.168.101.0/16	198.0.0.0/24	7
192.168.101.0/32	198.0.0.0/24	3 <-- Second Longest Match

Although this is not how the search is implemented, consider that all the records matching the server and local DNS IP address are gathered into a set. The records in this set are sorted in descending order first by local DNS mask, and then by server mask. The highest record in the sorted set determines which is the shortest path between the client and a virtual server. For example, if the third list record in the above example is removed, then the first and fifth records tie for longest match on local DNS, but the fifth wins because it has the more specific server mask.

Implementing topology-based access control

Any server/local DNS matching a list record with a score below the **acl_threshold** is interpreted as if the virtual server were unavailable. For example, if a local DNS **198.0.0.0** requests a name resolution, any virtual server owned by BIG/ip Controller **192.168.101.0** is considered **down** for load balancing purposes due to the first list entry. This provides a hook for an administrator to set up access control to data centers based on local DNS IP address.

Explicitly allowing or denying access

You may want to define a wildcard list record that you can use to prevent users from being locked out when access control is turned on (when the **acl_threshold** is set to a value greater than zero). If the 3DNS Controller compares the local DNS server's IP address to the specific list records but does not find a match, it can use a wildcard list record to determine how to handle the resolution request.

A wildcard list record is the last list record in the topology statement and uses the following syntax:

```
0.0.0.0/0      0.0.0.0/0 <score>
```

By using the subnet bitmask values **0** in the wildcard list record, this record will always be chosen last by the longest match rule.

The **<score>** parameter setting either allows or denies access, depending on whether its value is set greater than or less than the **acl_threshold** setting. A **<score>** value that is greater than or equal to the **acl_threshold** setting allows access. A **<score>** value that is less than the **acl_threshold** setting denies access.

If no wildcard list record is provided, the following is assumed:

```
0.0.0.0/0      0.0.0.0/0      0
```

Using access control to limit path probing

The **limit_probes** parameter specifies whether to apply access control to the probing of paths. If this parameter is set to **yes**, the 3DNS Controller requests a given BIG/ip Controller to probe only those local DNS servers that can connect to it according to the **probe_threshold** value and the topology map scores. In the example **topology** statement that starts on page 6 -19, the 3DNS Controller does not send a local DNS **200.0.0.0** to the BIG/ip Controller **192.168.101.0** for probing, but would send it to the BIG/ip Controller **192.168.103.0**.

To configure topology-based probe access control using the F5 Configuration utility

1. In the navigation pane, click **Topology**.
2. On the toolbar, click **Topology Settings**.
3. In the **ACL Threshold** box, specify a score threshold. Any list record (prober IP address/local DNS IP address) score that is less than the threshold value will not have access to the listed virtual servers.

4. In the **Probe Threshold** box, specify a threshold for probing. Any list record (prober IP address/local DNS IP address) score that is less than the probe threshold value prevents the local DNS from being probed by the specific prober. You can enter a value between 1 and 4294967295.
5. In the **Limit Probes** box, specify whether to apply access control to the probing of paths.
6. In the **Longest Match** box, specify whether the 3DNS Controller selects the record that is most specific, and thus has the longest mask, in cases where there are several IP/mask items that match a particular IP address.
7. Click **Update**.
8. On the toolbar, click **Add Virtual Servers**.
The Add Virtual Server to Topology screen opens.
9. In the **Virtual Server/Mask** boxes, type the virtual server's IP address in the first box, and the mask in the second box. These values combine to provide one of two values used to determine the longest match.
10. In the **LDNS/Mask** box, type the local DNS server's IP address in the first box, and the mask in the second box. These values combine to provide one of two values used to determine the longest match.
11. In the **Score** box, type the mask score, which is used for the comparison of virtual servers.
12. Click **Add**.

Repeat steps 9 through 12 for each virtual server that is to be used in the topology configuration.

To configure topology-based probe access control manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Place the **topology** statement at the end of the **wideip.conf** file.

3. Use the syntax shown in Figure 6.9 to define **topology** statement. Also see the example that starts on page 6 -19.

```
topology {  
  acl_threshold    <0..4294967295>  
  probe_threshold  <0..4294967295>  
  limit_probes     <yes | no>  
  longest_match    <yes | no>  
  <server cidr> <LDNS cidr> <score>  
}
```

Figure 6.9 Syntax for topology statement

An example configuration for topology access control

Suppose that your company maintains Spanish web sites. You have data centers in New York, Los Angeles, and Tokyo. You prefer that resolution requests made from clients located in North America are resolved by North American data centers. However, you do not mind if a few requests are sent to Tokyo when requests cannot be resolved in New York or Los Angeles.

Because of cost issues, you do not want requests from clients in South America to be sent to the New York data center. To achieve this, you can configure the **topology** statement as shown.

```
topology {
    acl_threshold 5
    probe_threshold 5
    limit_probes yes
    longest_match yes

    // server/mask    ldns/mask    score

    //////////////////////////////////////
    // North American LDNS's:
    //   198.0.0.0/8
    //   199.0.0.0/8

    // North America Priority List
    //
    // 1. New York
    // 2. L.A.
    // 3. Tokyo

    // New York
    192.168.101.0/24    198.0.0.0/8    30
    192.168.101.0/24    199.0.0.0/8    30

    // Los Angeles
    192.168.102.0/24    198.0.0.0/8    20
    192.168.102.0/24    199.0.0.0/8    20

    // Tokyo
    192.168.103.0/24    198.0.0.0/8    10
    192.168.103.0/24    199.0.0.0/8    10
```

Figure 6.10 Example syntax for the topology statement (continued on next page)

```

////////////////////
// South American LDNS's:
//   200.0.0.0/8
//   201.0.0.0/8

// South America Priority List
//
// 1. Tokyo
// 2. L.A.
// (New York excluded by acl_threshold)

// Tokyo
192.168.103.0/24    200.0.0.0/8    30
192.168.103.0/24    201.0.0.0/8    30

// Los Angeles
192.168.102.0/24    200.0.0.0/8    20
192.168.102.0/24    201.0.0.0/8    20

// New York
192.168.101.0/24    200.0.0.0/8     0
192.168.101.0/24    201.0.0.0/8     0

////////////////////
// Wildcard List Record
//
// By default, if a list record is not found in the
// topology map for an LDNS, the score is assumed to
// be 0. By including the following "wildcard" list
// record, all other LDNS's (not North or South America
// as specified above) are assigned a score of 1 so
// the acl_threshold does not indicate that the
// virtual servers are down.

0.0.0.0/0    0.0.0.0/0    1

}

```

Figure 6.11 Example syntax for the topology statement (continued from previous page)

Using the topology load balancing mode

The topology load balancing mode distributes connections based on the proximity of a local DNS to a particular data center. Proximity is determined by network IP addresses of the local DNS compared to that of the data centers, and not necessarily by geographical location.

Configure topology access control before attempting to use the Topology load balancing mode. If the topology list records are empty the virtual servers will be load balanced as Global Availability.

To implement the topology load balancing mode using the F5 Configuration utility

1. In the navigation pane, click **Wide IPs**.
2. Click a wide IP name.
The Edit Wide IP screen opens.
3. On the toolbar, click **Modify VSb Pool** or **Modify VSh Pool**, depending on which type of pool you have.
4. Select topology as the preferred, alternate, or fallback load balancing mode.
5. Click **Update** for the changes to take effect.

To implement the topology load balancing mode manually

1. On the 3DNS Maintenance menu, select **Edit 3DNS Configuration** to open the **wideip.conf** file.
2. Locate the **wideip** statement you want to edit.
3. Define topology as the preferred, alternate, or fallback load balancing mode.

Figure 6.12 shows a sample **wideip** definition where topology is the preferred load balancing mode.

```
wideip {  
  address 192.168.103.60  
  port 80  
  name "ntp.wip.domain.com"  
  pool {  
    name "poolA"  
    preferred topology  
    alternate rtt  
    address 192.168.101.60 // New York  
    address 192.168.102.60 // Los Angeles  
    address 192.168.103.60 // Tokyo  
  }  
}
```

Figure 6.12 Specifying load balancing according to topology score



7

Monitoring and Administration

- **Monitoring and administration utilities provided on the 3DNS Controller**
- **Working with the 3DNS Maintenance menu**
- **Controlling network traffic patterns with production rules**
- **Working with the production rules scripting language**
- **Setting up the 3DNS SNMP agent**
- **Changing passwords for the 3DNS Controller**
- **Viewing system statistics**

Monitoring and administration utilities provided on the 3DNS Controller

The 3DNS Controller provides utilities for monitoring and administration of the 3DNS Controller. You can monitor system statistics for all components of your 3DNS Controller, as well as perform configuration tasks.

The 3DNS Controller provides the following configuration, monitoring, and administration utilities:

❖ **F5 Configuration utility**

The F5 Configuration utility is a browser-based application that you can use to configure and monitor the 3DNS Controller. You may have used the F5 Configuration utility to define your network setup. The F5 Configuration utility supports Netscape Navigator, version 4.5, and Internet Explorer, version 4.x.

❖ **3DNS Maintenance menu**

Use the 3DNS Maintenance menu to simplify certain tasks such as starting the **big3d** agent and editing the **wideip.conf** file.

Working with the 3DNS Maintenance menu

The 3DNS Maintenance menu is a tool you can use to manually configure and monitor the 3DNS Controller only if you do not work with either the browser-based Configuration utility or the NameSurfer application.

You can use the 3DNS Maintenance menu to perform the following types of manual configuration tasks:

- ❖ Configure wide IPs
- ❖ View statistics
- ❖ Work with the **big3d** agent
- ❖ Manage synchronized files
- ❖ Work with security issues
- ❖ Configure the 3DNS web server

❖ Work with **syncd**

Figure 7.1 shows the 3DNS Maintenance menu:

```
3 D N S(®)  Maintenance Menu

Generate RSA Authentication
Generate and Copy iQuery Encryption Key
Check versions of named, BIG/ip kernel and needed big3d
Edit big3d matrix
Install and Start big3d
Edit BIND Configuration
Edit 3DNS Configuration
Synchronize Metrics Data
Check big3d
Restart big3d
Reconfigure 3DNS Web Administration
Restart 3DNS Administration
Change/Add Users for 3DNS Web Administration
Dump and List named Database
Stop syncd
Restart syncd
Checkpoint synced files
Rollback checkpoint
Enter 'q' to Quit
```

Figure 7.1 3DNS Maintenance menu

To use the 3DNS Maintenance menu

1. Type the following command to open the menu:
3dnsmaint
2. Select the command to execute, then press the Enter key.

Each command is described in the sections that follow.

Configuring wide IPs

You can use the following commands to edit BIND and 3DNS Controller configuration files. However, we recommend that you use NameSurfer to handle BIND Configuration, and that you use the F5 Configuration utility to configure wide IPs.

Edit BIND Configuration

The **Edit BIND Configuration** command opens the **named.conf** file for editing. You should only use this command if you are performing all configuration tasks manually. It is important that you do not use this command if you are using the F5 Configuration utility or NameSurfer.

Edit 3DNS Configuration

The **Edit 3DNS Configuration** command runs the **edit_wideip** script, which performs the following tasks:

- ❖ Opens the **wideip.conf** file for editing.
- ❖ Copies the **wideip.conf** file to all other 3DNS Controllers in the local 3DNS Controller's sync group.
- ❖ Restarts **named**.

Viewing statistics

You can use the following command to view various 3DNS Controller statistics. For more statistics information, you can also use the Statistics area of the F5 Configuration utility (as described earlier in this chapter).

Dump and List named Database

The **Dump and List named Database** command corresponds to the **3dprint** script, which lets you view these statistics screens on the command line:

❖ **sum**

Displays summary statistics, such as the 3DNS Controller version, the total number of resolved requests, and the load balancing methods used to resolve requests.

❖ **paths**

Displays path statistics, such as round trip time and packet completion rate.

❖ **ldns**

Displays statistics collected for local DNS servers, including the number of resolution requests received from a given server, and the current protocol used to probe the server.

❖ **vs**

Displays statistics about BIG/ip and host virtual servers, such as the server state, and the number of times it has received resolution requests.

❖ **bigips**

Displays statistics about all BIG/ip Controllers known to the 3DNS Controller, including the number of virtual servers each BIG/ip Controller manages, and the number of times that the 3DNS Controller resolves requests to those virtual servers.

❖ **hosts**

Displays statistics about all hosts known to the 3DNS Controller, including the number of times that the 3DNS Controller resolves requests to the host.

❖ **wips**

Displays statistics about each wide IP defined on the 3DNS Controller, including load balancing information and the remaining time to live before the wide IP's metrics data needs to be refreshed.

Working with the big3d agent

You can use the following commands to work with the **big3d** agent, which collects information about paths between a data center and a specific local DNS server.

Check versions of named, BIG/ip kernel and needed big3d

The **Check versions of named, BIG/ip kernel and needed big3d** command runs the **big3d_version** script, which displays version numbers for all BIG/ip Controllers known to the 3DNS Controller, as well as the version numbers of the **big3d** agent and **named** utility running on each BIG/ip Controller.

Edit big3d matrix

The **Edit big3d matrix** command opens for editing a file that lists version numbers for all BIG/ip Controllers known to the 3DNS Controller and the version numbers of the **big3d** agent and **named** utility running on each BIG/ip Controller.

You do not need to edit this file unless a new BIG/ip kernel or a **named** version create a conflict. If this happens, you need to place a new version of the **big3d** agent on all BIG/ip Controllers.

The **Install and Start big3d** command uses the matrix file to determine which version of the **big3d** agent to transfer.

Install and Start big3d

The **Install and Start big3d** command runs the **big3d_install** script, which installs and starts the appropriate version of the **big3d** agent on each BIG/ip Controller in the network.

Check big3d

The **Check big3d** command runs the **big3d_check** script, which verifies that each BIG/ip Controller is running the **big3d** agent.

Restart big3d

The **Restart big3d** command runs the **big3d_restart** script, which stops and restarts the **big3d** agent on each BIG/ip Controller.

Managing synchronized files

You can use the following commands to copy metrics data to a new 3DNS Controller, archive synchronized files, or retrieve an archive.

Synchronize Metrics Data

The **Synchronize Metrics Data** command runs the **3dns_sync_metrics** script, which prompts you to either copy metrics data from the local 3DNS Controller to other 3DNS Controllers, or from a remote 3DNS Controller to the local 3DNS Controller.

You should only use this command when you are configuring a new 3DNS Controller.

Checkpoint synced files

The **Checkpoint synced files** command runs the **syncd_checkpoint** script, which creates a *checkpoint file*. A checkpoint file is a compressed tar file that contains an archive of the files that are synchronized.

For more information, see *syncd_checkpoint*, on page B-38.

Rollback checkpoint

The **Rollback checkpoint** command runs the **syncd_rollback** script, which unrolls a checkpoint file.

For more information, see *syncd_rollback*, on page B-39.

Working with security issues

You can use the following menu commands to address security issues for your network setup:

- ❖ Generate RSA Authentication
- ❖ Generate and Copy Encryption iQuery Key

Generate RSA Authentication

The **Generate RSA Authentication** command runs the **3dns_auth** script, which generates a password authentication by setting the RSA Authentication parameter to **yes** in **/etc/sshd_config.conf** and copying the **ssh** key to each 3DNS Controller and BIG/ip Controller. When prompted for an RSA passphrase, press the Enter key instead of typing a password.

For more information, see *3dns_auth*, on page B-31.

Generate and Copy Encryption iQuery Key

The **Generate and Copy Encryption iQuery key** command runs the **install_key** script, which then runs the **F5makekey** script. **F5makekey** generates a seed key for encrypting communications between the 3DNS Controller and BIG/ip Controller.

For more information, see *install_key* and *F5makekey*, on page B-37.

◆ Note

This command is not available in the international version of 3DNS Controller.

Using the 3DNS web server

You can use the following commands to configure the 3DNS web server.

Reconfigure 3DNS Web Administration

The **Reconfigure 3DNS Web Administration** command runs the **3dns_web_config** script, which lets you make configuration changes to the 3DNS web server.

Restart 3DNS Administration

The **Restart 3DNS Administration** command runs the **3dns_admin_start** script, which restarts the 3DNS web server.

Change/Add Users for 3DNS Web Administration

The **Change/Add Users for 3DNS Web Administration** command runs the **3dns_web_passwd** script, which lets you provide restricted or administrative access to the 3DNS web server for selected users only, and assigns passwords for those users. Users with restricted access have access to the statistics area only. Users with administrative access have access to all areas of the 3DNS web server. If you do not use this script, no user has access to the 3DNS web server.

◆ Note

*The **3dns_web_passwd** script is run by the First-Time Boot utility.*

Working with syncd

You can use the following commands to work with **syncd**, the synchronization daemon that runs on all 3DNS Controllers. The function of **syncd** is to update and synchronize all 3DNS Controller configuration files.

Stop syncd

The **Stop syncd** command runs the **syncd_stop** script, which stops the **syncd** daemon, if it is running.

Restart syncd

The **Restart syncd** command runs the **syncd_start** script, which restarts the **syncd** daemon if it is already running, or starts it if it is not.

Controlling network traffic patterns with production rules

Production rules are a policy-based management tool that you can use to dynamically change how the 3DNS Controller distributes connections across the network. You can also use production rules to send system administrators notifications of specific events. Production rules are based on triggers, such as time of day, current traffic patterns, or current traffic volume. For example, you can configure a production rule that changes the load balancing mode to QOS during your peak business hours, and you can configure a production rule that notifies you when the number of name resolution requests exceeds a specific number.

You can create production rules that apply to the system in general, or you can create production rules for specific wide IPs.

If you want to configure basic production rules, we recommend that you use the F5 Configuration utility. If you want to create custom production rules, you should review the following section, *Working with the production rules scripting language*, on page 7-15, which describes the scripting language you use to configure production rules manually. You may also want to contact a technical support engineer for additional assistance with complex configurations.

Setting up production rules in the F5 Configuration utility

The F5 Configuration utility uses a wizard-style format to help you set up production rules. The screen prompts that you see during the configuration process vary, depending on the items you select in each screen. However, to configure any production rule, you essentially perform three basic steps:

- ❖ **Define the type of rule**

There are two types of rules: global production rules and wide IP production rules.

- ❖ **Define the rule trigger**

There are two types of rule triggers: a set time or time interval, and specific system events.

❖ **Defining the action taken**

There are two basic types of rule actions: sending user-definable messages to log files or email accounts, and changing specific load balancing settings.

The following sections discuss each production rule option in detail, and should provide you all of the information you need to complete the production rule wizard.

Viewing, adding, and deleting production rules

When you click Production Rules in the Configuration utility, the Production Rules wizard screen opens. The screen displays the list of existing global and wide IP production rules. You can add a new rule by clicking the **Add Production Rule** toolbar button, which actually starts the production rule wizard. The wizard prompts you to specify the various production rule options, and then allows you to review your selections before you save the production rule to the configuration.

Note that you can modify existing production rules by clicking the rule name in the list, and you can delete a production rule at any time by clicking the trash can icon next to the rule name.

Choosing the rule type

The first step in the production rule wizard is to choose whether the production rule is a global production rule or a wide IP production rule.

❖ **Global production rules**

Global production rules send messages to log files or to specific email accounts, based on a set time interval or on standard events. The standard events are listed and described in the following section.

❖ **Wide IP production rules**

Wide IP production rules are based either on the time of day, or on standard events. Wide IP production rules can change the current load balancing modes for the preferred, alternate, or fallback methods, they can reconfigure ratio settings for

individual virtual servers, and they can reconfigure the coefficients for Quality of Service mode. Wide IP production rules can also send messages to log files or email accounts.

After you choose a rule type, the wizard prompts you to name the rule and allows you to add a brief description of the rule.

Defining time-based triggers

The next step in the wizard prompts you to choose a trigger for the production rule. There are two basic types of triggers that you can set up: time-based triggers and event-based triggers. This section describes the options for the time-based triggers, and the following section describes options for the event-based triggers. Once you review the information for the type of trigger you want to set up, you can skip to the section about choosing an action on page 7 -14.

Time-based triggers include two types. Global production rules trigger on set time intervals, while wide IP production rules trigger at specific times on specific days. To set a time interval for a global production rule, you simply define the number of seconds that elapse between each action the production rule executes.

A wide IP production rule can trigger at a specific time of day, on a specific day of the week, on a specific date, or at a specific time on a specific date. The following procedures explain how to set up each type of time trigger for wide IP production rules.

To apply a time of day variable

1. From the Time Variable table, select **Time**.
2. From the **Start Time, Hour** box, select the hour you want the production rule action to begin.
3. From the **Start Time, Minutes** box, select the minute you want the production rule action to begin.
4. From the **Stop Time, Hour** box, select the hour you want the production rule action to stop.
5. From the **Stop Time, Minutes** box, select the minute you want the production rule action to stop.

Once you define the time of day that triggers the production rule, you continue with the wizard and begin to define the production rule action.

To apply a day of the week variable

1. From the Time Variable table, select **Day**. A table appears from which you select the day to start and stop the action.
2. From the **Start Day** box, select the day you want the production rule action to begin.
3. From the **Stop Day** box, select the day you want the production rule action to stop.

Once you define the day of the week that triggers the production rule, you continue with the wizard and begin to define the production rule action.

To apply a date variable

1. From the Time Variable table, select **Date**. A table opens from which you select the date to start and stop the action.
2. From the **Start Date** box, type the date you want the production rule action to begin (mm/dd/yyyy).
3. From the **Stop Date** box, type the date you want the production rule action to stop (mm/dd/yyyy).

Once you define the date that triggers the production rule, you continue with the wizard and begin to define the production rule action.

To apply a combined date and time variable

1. From the Time Variable table, select **Date/Time**. Two tables open where you select the start and stop dates and times.
2. From the **Start Date** box, type the date you want the production rule action to begin (mm/dd/yyyy).
3. From the **Stop Date** box, type the date you want the production rule action to stop (mm/dd/yyyy).

4. From the **Start Time, Hour** box, select the hour you want the production rule action to begin.
5. From the **Start Time, Minutes** box, select the minute you want the production rule action to begin.
6. From the **Stop Time, Hour** box, select the hour you want the production rule action to stop.
7. From the **Stop Time, Minutes** box, select the minute you want the production rule action to stop.

Once you define the date and time that triggers the production rule, you continue with the wizard and begin to define the production rule action.

Defining event-based triggers

Both global production rules and wide IP production rules can trigger on standard events, such as when a name resolution process begins. Wide IP production rules support two additional types of event-based triggers. You can set a wide IP production rule to trigger when a specific local DNS server makes a name resolution request, or to trigger when a user-specified number of name resolution requests are received by the 3DNS Controller.

The list of standard events that can trigger both global and wide IP production rules includes the following:

❖ **ResolveNameBegin**

The production rule takes action each time the 3DNS Controller receives a new resolution request.

❖ **ResolveNameEnd**

The production rule takes action each time the 3DNS Controller completes a name resolution.

❖ **FallbackToStatic**

The production rule takes action each time the fallback load balancing method is used in a wide IP.

❖ **SIGINT**

The production rule takes action each time the 3DNS Controller receives a SIGINT command.

❖ **SIGHUP**

The production rule takes action each time the 3DNS Controller receives a SIGHUP command.

❖ **ReapPaths**

The production rule takes action each time the 3DNS Controller reaps obsolete path information.

❖ **CRC_Failure**

The production rule takes action each time iQuery communications on the 3DNS Controller experiences a CRC failure.

❖ **DownServer**

The production rule takes action each time the 3DNS Controller detects that another 3DNS Controller, BIG/ip Controller, or host server becomes unavailable.

❖ **DownVS**

The production rule takes action each time the 3DNS Controller detects that a virtual server becomes unavailable.

❖ **DoneINT**

The production rule takes action after the wideip.conf file is read on startup (a one-time event).

❖ **DoneConfigFile**

The production rule takes action each time the 3DNS Controller configuration is re-read (when an **ndc reload** command is issued, for example).

Choosing the action

After you specify the production rule trigger, the wizard prompts you to choose the action that the production rule takes. Note that the actions that a production rule can take depend in part on whether the production rule is a global rule or a wide IP rule. For example, both global production rules and wide IP production rules can send user-defined messages to log files, or to specific email accounts, but only wide IP production rules can alter load balancing modes.

❖ **Sending user-defined messages**

Both global and wide IP production rules can send user-defined messages to the syslog file, or to a specific email account.

❖ Changing the load balancing mode settings

Wide IP production rules can change load balancing mode settings for the wide IP. You can change the preferred, alternate, and fallback methods, and you can change QOS coefficient settings.

❖ Changing virtual server ratios

You can change virtual server ratios to alter the distribution load when the load balancing mode is set to Ratio.

❖ Specifying a virtual server to return

You can specify that the 3DNS Controller return a specific virtual server, rather than choosing a virtual server using load balancing.

Once you specify an action, the production rules wizard prompts you to review all of the production rule settings, and then saves the production rule to the configuration.

Working with the production rules scripting language

The production rules scripting language uses constructs and statements that are similar in syntax to Perl script and the C programming language. If you have a good working knowledge of Perl or C, you may want to create your own custom production rules using the guidelines in this section in conjunction with the examples provided both here and in the sample **wideip.conf** file (installed on the 3DNS Controller and also available in Appendix A).

If you need to add custom production rules to your configuration, but you do not want to work out the implementation yourself, you can contact a professional services representative for assistance.

Inserting production rules in the wideip.conf file

Production rules are part of the **wideip.conf** file, and you can either insert them directly in the file, or you can store them in a separate file and include them by reference. If you want to use the Configuration utility to manage the 3DNS Controller configuration, you must store manually configured production rules in a separate file and include them by reference. If you attempt to use custom production rules in a file that you edit using the F5 Configuration utility, the production rule syntax may become corrupt.

If you include custom production rules directly in the **wideip.conf** file, you must manually edit and maintain the **wideip.conf** file; you cannot use the F5 Configuration utility for configuration administration.

Execution and management of production rules

The **3dscrip** utility manages and executes production rules according the following guidelines:

- ❖ **3dscrip** supports conditional execution of production rules using the **if** statement. You can use **if** statements for wide IP production rules, and in global production rules only if they are embedded within a **when** or an **every** statement.
- ❖ **3dscrip** supports event-driven execution of production rules using the **when** statement. You can use the **when** statement only in global production rules.
- ❖ **3dscrip** supports periodic execution of production rules using the **every** statement. You can use the **every** statement only in global production rules.
- ❖ Each production rule is uniquely identified by a label.
- ❖ Each production rule can be deleted using its label.
- ❖ All production rules at the global scope can be deleted.
- ❖ All production rules at the wideip-pool scope can be deleted.
- ❖ Each production rule can be replaced.
- ❖ Each production rule can be annotated with a character string.

The if statement

The **if** statement is a standard statement which defines an event condition that triggers a production rule action. Typically you use **if** statements in wide IP production rules. An **if** statement must adhere to the following guidelines:

- ❖ The **if** statement can be specified in the scope of a wide IP **pool** statement.
- ❖ The **if** statement can be nested in another **if** statement.
- ❖ Multiple **if** statements can be specified in the same scope.
- ❖ Nesting **if** statements is unlimited except by the memory capacity of the 3DNS Controller.
- ❖ The first form of an **if** statement:

```
if(conditional-expression) { <action> ... }
```

- ❖ The second form of an **if** statement:

```
if(conditional-expression) { <action> ... } else { <action> ... }
```

- ❖ The conditional-expression is composed of one of:
  - A primitive-expression
  - A primitive-expression followed by a relational-operator followed by a primitive-expression
  - A primitive-expression followed by an arithmetic-operator followed by a primitive-expression
  - Two conditional-expressions joined by a logical-operator
- ❖ The primitive-expression can be one of:
  - A keyword which is evaluated when the conditional expression is evaluated
  - An intrinsic function which is evaluated when the conditional expression is evaluated
  - A literal value enclosed in full quotes
  - A conditional-expression enclosed in parenthesis
  - A unary-operator followed by a conditional-expression enclosed in parenthesis
- ❖ A logical-operator is one of:

- `||` (logical OR)
- `&&` (logical AND)
- ❖ A relational-operator is one of:
  - `==` (equality)
  - `!=` (not equal)
  - `>` (greater than)
  - `>=` (greater than or equal to)
  - `<` (less than)
  - `<=` (less than or equal to)
- ❖ An arithmetic-operator is one of:
  - **`mod`** (modulus)
- ❖ A unary operator is one of:
  - `!` (unary negation)
  - `-` (unary minus)
- ❖ A keyword is one of:
  - **`day`**
  - **`time`**
  - **`date`**
  - **`datetime`**
  - **`ldns_ip`**
  - **`wip_ip`**
  - **`wip_name`**
  - **`wip_num_resolves`**
  - **`preferred`**
  - **`alternate`**
  - **`fallback`**
  - **`rtt`**
  - **`completion_rate`**
  - **`hops`**
  - **`packet_rate`**

- **topology**
- ❖ An intrinsic function is one of:
  - **isLdnsInNet(ip, mask)**
  - **isLdnsInAS(ip, mask)**
- ❖ The precedence of logical, relational, and unary operators is the same as in ANSI-c.

The when statement

The **when** statement is a standard statement which defines a specific event condition that triggers a production rule action. A **when** statement can be used only in global production rules, and it must adhere to the following guidelines:

- ❖ The **when** statement can be specified at the top scope of **wideip.conf**, after the **wideip** definition(s) and before the **topology** statement.
- ❖ Multiple **when** statements can be specified in the same scope.
- ❖ Nesting of **when** statements is not allowed.
- ❖ The form of a **when** statement:
when(event) { <action> ... }
- ❖ An event can be one of the following (see page 7 -13 for detailed descriptions of each event):
  - **ResolveNameBegin**
  - **ResolveNameEnd**
  - **FallbackToStatic**
  - **SIGINT**
  - **SIGHUP**
  - **SIGUSR1**
  - **SIGUSR2**
  - **SIGCHLD**
  - **ReapPaths**
  - **ReapLdns**

- **CRC_Failure**
- **DownServer**
- **DownVS**
- **DoneInit**
- **DoneConfigFile**

The every statement

The **every** statement is a standard statement which defines a time interval at which the production rule action triggers, such as every 60 seconds. An **every** statement can be used only for a global production rule, and it must adhere to the following guidelines:

- ❖ The **every** statement can be specified at the top scope of **wideip.conf**, after the **wideip** definition(s) and before the **topology** statement.
- ❖ Multiple **every** statements can be specified in the same scope.
- ❖ Nesting of **every** statements is not allowed.
- ❖ The form of the **every** statement:

every(<seconds>) { <action> ... }

Production rule actions

The production rules language supports the following actions. Not all actions apply to all production rule types. For example, the actions that change load balancing settings are valid only for wide IP production rules. Actions such as defining a log string can be used in either global production rules or wide IP production rules. Each action below specifies which production rule types can use it.

- ❖ **preferred <lbmode>**

This action changes the preferred load balancing method in a wide IP. You can use this action only in a wide IP production rule.

❖ **alternate <lbmode>**

This action changes the alternate load balancing method in a wide IP. You can use this action only in a wide IP production rule.

❖ **fallback <lbmode>**

This action changes the fallback load balancing method in a wide IP. You can use this action only in a wide IP production rule.

❖ **log(<string>)**

This action sends the specified string to the syslog utility, which writes the string to the syslog file. You can use this action in either a wide IP production rule or a global production rule.

❖ **log2mail(<string>)**

This action sends the specified string to the Sendmail utility, which creates a mail message and forwards it to the administrative email account specified for Sendmail (see *log2mail* on page B -13 for details about log2mail syntax). You can use this action in either a wide IP production rule or a global production rule.

❖ **vs(<ip>:<port>).ratio <n>**

This action changes the ratio setting for a specific virtual server in a wide IP pool. You can use this action only in a wide IP production rule.

❖ **return_vs(<ip:port>)**

This action skips the load balancing process and instead returns the specified virtual server to the requesting client. You can use this action only in a wide IP production rule.

Production rule examples

The following examples offer a variety of custom production rules that you may want to implement or expand on for your own network. Other production rule examples are included in the sample **wideip.conf** file installed on the 3DNS Controller (and available in Appendix A).

Load balancing according to time of day

you can set up production rules, ahead of time, to deal with future needs and client demands for events. For example, say your company has a software distribution scheduled for release next Tuesday at 5:00 p.m. Pacific Standard Time. The new software will be available for download from the FTP sites at that time, and you expect that during the first week, traffic will be 10 times normal, with frequent bursts during standard work hours, 7 a.m. to 6 p.m. However, the client base spans four time zones with an FTP server farm on the east coast in New York (**192.168.101.50**), and another on the west coast in Los Angeles (**192.168.102.50**). The 3DNS Controller is located on the east coast and runs on Eastern Standard Time. You are willing to accept some network latency in return for guaranteed connections.

Figure 7.2 shows a sample production rule that handles the connections according to anticipated load at specific times of the day.


```
wideip {  
  address 192.168.101.50:21  
  name "ftp.domain.com"  
  pool {  
    preferred ratio  
    address 192.168.101.50 ratio 2  
    address 192.168.102.50 ratio 1  
    rule "ftp_balance"  
      // Night time: qos  
      if(time > "21:00" && time < "07:00") {  
        preferred leastconn  
      }  
      else {  
        preferred ratio  
        // East Coast  
        rule "east" if(time < "10:00") {  
          vs.(192.168.101.50).ratio 3  
          vs.(192.168.102.50).ratio 1  
        }  
        // Both coasts are at peak demand  
        else {  
          rule "both" if(time < "18:00") {  
            vs.(192.168.101.50).ratio 1  
            vs.(192.168.102.50).ratio 1  
          }  
          // West Coast  
          else {  
            vs.(192.168.101.50).ratio 1  
            vs.(192.168.102.50).ratio 3  
          }  
        }  
      }  
    }  
  }  
}
```

Figure 7.2 Load balancing by time of day

Load balancing according to LDNS

One interesting application of production rules is when you create a rule that triggers based on a specific local DNS server making a name resolution request. The following example is based on a web site published in three languages: English, Spanish, and Japanese. Suppose that the addresses in the network **10.10.0.0** are allocated to Japanese speakers, and the addresses in the network **10.11.0.0** are allocated to Spanish speakers. The production rule shown in Figure 7.3 uses the address of the requesting LDNS to determine which virtual server should receive the connection.

```
wideip {
  address 192.168.101.50:80
  name "www.domain.com"
  pool {
    rule "Japanese" if(isLdnsInNet(10.10.0.0, 255.255.0.0)) {
      return_vs(192.168.103.50:80)
    }
    else {
      rule "Spanish" if(isLdnsInNet(10.11.0.0, 255.255.0.0)) {
        return_vs(192.168.102.50:80)
      }
      else { // assume English
        return_vs(192.168.101.50:80)
      }
    }
  }

  address 192.168.101.50 // English
  address 192.168.102.50 // Spanish
  address 192.168.103.50 // Japanese
}
```

Figure 7.3 Load balancing by IP address of local DNS

Hacker detection

Another interesting example of triggering a production rule based on the requesting local DNS server is to take evasive action against known hackers attempting to access your system. The production

rule shown in Figure 7.4 sends the hacker to a special server, rather than flat out rejecting the connection. As an alternative, you could change the rule to return a non-routable or non-existent address.

```
when(ResolveNameBegin) {  
  rule "roach_motel" if(isLdnsInNet(10.20.30.4, 255.255.255.0)) {  
    // Send this guy to our "roach motel" for hackers.  
    // This address doesn't need to be listed in any wideip pool.  
    // It's reserved for us to watch hackers under the microscope.  
    log2mail("Hacker $ldns_ip came back")  
    return_vs(192.168.1.46:80)  
  }  
}
```

Figure 7.4 Sending a hacker to a specific server

A related example, shown in Figure 7.5, illustrates a production rule that deals with attacks against iQuery communications. The production rule would warn you if the 3DNS Controller detected a hack attempt against the iQuery protocol, based on a communication failure.

```
Rule "iQuery_hacked" when(CRC_Failure) {  
  log2mail("Got CRC Failure")  
}
```

Figure 7.5 Detecting an iQuery failure due to potential attack

Setting up the 3DNS SNMP agent

This section describes the management and configuration tasks for the simple network management protocol (SNMP) agent and management information bases (MIBs) available with the 3DNS Controller.

WARNING

You must configure the SNMP agent on the 3DNS Controller in order to monitor the 3DNS Controller using the F5 Networks see/IT Network Manager.

The 3DNS SNMP agent and MIBs allow you to manage the 3DNS Controller by configuring traps for the SNMP agent or polling the controller with your standard network management station (NMS).

You can configure the 3DNS SNMP agent to send traps to your management system with the F5 Configuration utility. You can also set up custom traps agent setup by editing several configuration files.

Security options are available that let you securely manage information collected by the 3DNS SNMP agent, including:

- ❖ Community names
- ❖ TCP wrappers
- ❖ View access control mechanism (VACM)

Downloading the MIBs

The 3DNS Controller includes a private 3DNS SNMP MIB. This MIB is specifically designed for use with the 3DNS Controller. You can configure the SNMP settings in the F5 Configuration utility, or on the command line.

SNMP management software requires that you use the MIB files associated with the device. You may obtain two MIB files from the 3DNS directory **/usr/contrib/f5/mibs**, or you can download the files from the **Additional Software Downloads** section of the F5 Configuration utility home page.

❖ 3dns.my

This is a vendor MIB that contains specific information for properties associated with specific F5 functionality, such as load balancing.

❖ rfc1611.my

This is a DNS server MIB (RFC 1611) that provides standard management information.

For information about the objects defined in **3dns.my**, refer to the descriptions in the object identifier (OID) section of the MIB file. For information about the objects defined in **rfc1611.my**, refer to RFC 1611.

Understanding configuration file requirements

You need to make changes to several configuration files on the 3DNS Controller before you use the SNMP agent. Once you change these configuration files, you need to restart the SNMP agent.

`/etc/hosts.deny`

This file must be present to deny by default all UDP connections to the SNMP agent. The contents of this file are as follows:

ALL : ALL

`/etc/hosts.allow`

The **`/etc/hosts.allow`** file specifies the hosts that are allowed to access the SNMP agent. You can configure access to the SNMP agent with the **`/etc/hosts.allow`** file in one of two ways. You can either type in an IP address, or list of IP addresses, that are allowed to access the SNMP agent, or you can type in a network address and mask to allow a range of addresses in a subnetwork to access the SNMP agent.

For a specific list of addresses, type in the list of addresses you want to allow to access the SNMP agent. Addresses in the list must be separated by blank space or by commas. The basic syntax is as follows:

daemon: <IP address> <IP address> <IP address>

For example, by typing the following line, the SNMP agent accepts connections from the specified IP addresses:

snmpd: 128.95.46.5 128.95.46.6 128.95.46.7

For a range of addresses, the basic syntax is as follows, where **daemon** is the name of the daemon, and **NETWORKADDRESS/MASK** specifies the network that is allowed access:

daemon: NETWORKADDRESS/MASK

For example, the following line sets the **snmpd** daemon to allow connections from the **128.95.46.0/255.255.255.0** address:

snmpd: 128.95.46.0/255.255.255.0

The example above allows the 256 possible hosts from the network address **128.95.46.0** to access the SNMP daemon. Additionally, you may use the keyword **ALL** to allow access for all hosts or all daemons.

/usr/contrib/isode/etc/snmpd.rc

The **/usr/contrib/isode/etc/snmpd.rc** file controls most aspects of the SNMP agent. This file is used to set up and configure certain traps, passwords, and general SNMP variable names. A few of the necessary variables are listed below:

❖ System Contact Name

The System Contact is a MIB-II simple string variable defined by almost all SNMP boxes. It usually contains a user name, as well as an email address. This is set by the **variable syscontact** line.

❖ Machine Location (string)

The Machine Location is a MIB-II variable that almost all boxes support. It is a simple string that defines the location of the box. This is set by the **variable syslocation** line.

❖ Community String

The community string clear text password is used for basic SNMP security. This also maps to VACM groups, but for initial read/only access, it is limited to only one group.

❖ Trap Configuration

Trap configuration is controlled by these entries in the `/usr/contrib/isode/etc/snmpd.rc` file. Each line defines the three parameters for a trap:

trap <community> <trap sink> [trap port]

The **<community>** parameter specifies the password, the **<trap sink>** parameter specifies the IP address to which the trap is sent, and the **[trap port]** specifies the port on which the trap is received.

❖ System IP Setting

You must set the system IP address using the **sysip** command; if this setting is not present, the **checktrap.pl** script will fail to send all 3DNS-specific traps. Use the following syntax to set the system IP address:

sysip <3DNS IP address>

`/etc/rc.local`

The following entry in the `/etc/rc.local` file sets the SNMP agent to automatically start up when you boot the 3DNS Controller (Figure 7.6).

```
# 3DNS SNMP Agent
if [ -f /usr/contrib/isode/etc/snmpd.rc ]; then
    /sbin/snmpd -c /usr/contrib/isode/etc/snmpd.rc
fi
```

Figure 7.6 Starting the SNMP agent in the `/etc/rc.local` file.

If the `/usr/contrib/isode/etc/snmpd.rc` file is present on your system, the SNMP agent starts automatically.

Syslog

You must configure **syslog** to send syslog lines to **checktrap.pl**. If the syslog lines match the specified regular expression in the **snmptrap.conf** file, the **checktrap.pl** script generates a valid SNMP trap. The following line in the **/etc/syslog.conf** file causes the **syslog** utility to send the specified log output to the **checktrap.pl** script. The **checktrap.pl** script then compares the logged information against the **snmptrap.conf** file to determine if a trap should be generated:

```
local2.* | exec /sbin/checktrap.pl.
```

Configuring the 3DNS SNMP agent settings

You can use the F5 Configuration utility to configure the following aspects of the 3DNS SNMP agent:

❖ **Client access**

You can define a network address and netmask for a workstation from which SNMP requests are acceptable.

❖ **System information**

You can name a system contact, a machine location, and a community string.

❖ **Trap configuration**

You can enter a trap sink and a trap community.

To set SNMP properties in the F5 Configuration utility

The F5 Configuration utility provides sample SNMP settings for your reference. If you want to use the 3DNS SNMP MIB, you need to replace these sample settings with settings appropriate to your environment and your specific SNMP management software.

1. Click **SNMP** in the navigation pane.
The SNMP Configuration screen opens.
2. In the 3DNS SNMP Configuration screen, check **Enabled** to allow access to the 3DNS SNMP agent.

3. In the **Allow Address** box, type the address, or addresses, of the management system from which the agent can accept requests. The addresses can be IP addresses, or network addresses. This setting restricts access to management information to a specific computer or computers running a management system. If you type in a list of addresses, type a comma after the last address.
4. In the **Allow Netmask** box, type the netmask for a range of IP addresses for machines from which the agent can accept requests.

Note that if you typed a list of IP addresses in the **Allow Address** box, or if you typed a network address in the **Allow Address** box, you should leave the **Allow Netmask** box blank.

5. In the **System Contact** box, type the contact name and email address for the person who should be contacted if the 3DNS Controller generates a trap.
6. In the **Machine Location** box, enter a machine location, such as *First Floor*, or *Building 1*, that describes the physical location of the 3DNS Controller.
7. In the **Community String** box, type a community name. The community name is a clear text password used for basic SNMP security and for grouping machines that you manage.
8. In the **Trap Sink** box, type the host that should be notified when a trap is sent by the 3DNS SNMP agent.
9. In the **Trap Community** box, type the community name to which this 3DNS Controller belongs. Traps sent from this box are sent to the management system managing this community.
10. Click **Update**.

Configuring options for the checktrap script

The **checktrap.pl** script reads a set of lines from standard input. The script checks each line against a set of regular expressions. If a line matches the regular expression, an SNMP trap is sent.

Options for checktrap

snmpd_conf_file=<snmp configuration file>

This is the file that contains the SNMP variables. The **checktrap.pl** script gets trap configuration information from this file. The default is **/usr/contrib/isode/etc/snmpd.rc**.

trapd_conf_file=<snmp trap configuration file>

This is the file that contains the regular expression to SNMP trap OID mappings. It also contains a description string that is added to the trap message. The default is **/etc/snmptrap.conf**.

trap_program=<snmp trap program>

This is the program that sends the trap. This program should be the **snmptrap** program included with the 3DNS Controller. The default is **/sbin/snmptrap**.

no_date_strip

This turns off automatic date stripping. Normally, each input line is expected to begin with a date. Typically, this date is stripped off before the trap is sent. This option keeps the date information in the trap. The date is stripped from the trap by default.

usage

Prints a usage string.

Changing passwords for the 3DNS Controller

The First-Time Boot utility prompts you to define a password that allows remote access to the 3DNS Controller, and also prompts you to define a password for the 3DNS Web server. You can change these passwords at any time.

To change the root user password for command line access

1. At the 3DNS Controller command line prompt, log in as root and use the **passwd** command.
2. At the **password** prompt, enter the password you want to use for the 3DNS Controller and press Enter.
3. To confirm the password, retype it and press Enter.

Changing passwords and adding new user IDs for the 3DNS web server

You can create new users for the 3DNS web server, change a password for an existing user, or recreate the password file altogether, without actually going through the 3DNS web server configuration process.

To add a new user ID using the F5 Configuration utility

1. In the navigation pane, click **User Admin**.
The User Administration screen opens.
2. In the **User Name** box, type the user ID to add.
The user name cannot include any of the following special characters: colon (:), single quote ('), double quote ("), plus sign (+), ampersand (&), pound sign (#), or percent sign (%).
3. Type the password that the user will use to access the 3DNS server in the **Password** box.
4. Confirm the password in the **Retype Password** box.

5. In the **Access Level** box, assign either **Read Only** or **Full Read/Write** access.
6. Click **Add**.
The new user ID appears in the **Current Users** table.

To change or add user information using the 3DNS Maintenance menu

Select the **Change/Add Users for 3DNS Web Administration** command.

To create new users and change passwords for existing users manually

The following command creates a new user ID, or changes the password for an existing user ID. In place of the **<username>** parameter, enter the user ID for which you want to create a password:

```
/var/f5/httpd/bin/htpasswd /var/f5/httpd/basicauth/users \  
<username>
```

Once you enter the command, you are prompted to enter the new password for the named user.

To create a new password file manually

The following command recreates the 3DNS web server password file, and defines one new user ID and password. In place of the **<username>** parameter, enter the user ID that you want to create:

```
/var/f5/httpd/bin/htpasswd -c /var/f5/httpd/basicauth/users \  
<username>
```

Once you enter the command, you are prompted to enter the new password for the named user.

Viewing system statistics

The 3DNS Controller lets you view current information about BIG/ip Controllers, other host machines, virtual servers, paths, and wide IPs in the configuration.

To view system statistics

1. From the F5 Configuration utility, click **Statistics**.
2. From the list, you can choose specific types of statistics, such as system statistics or virtual server metrics.

Viewing summary statistics

From the **Statistics** list, click **Summary** to view the following information about the 3DNS Controller system.

Note that you can configure how often to refresh the statistical information in all of the summary tables by changing the value in the **Refresh Interval** box and clicking **Refresh**.

General

Item	Description
3DNS Version	Displays the version number of the 3DNS Controller software.
Max Datasize	Displays the maximum amount of memory that is available for the 3DNS Controller to use.
Sync State	Indicates whether the 3DNS Controller is a principal or receiver.
Start Time	Displays the date and time that the system was booted.
Current Time	Displays the current date and time.
Last Reload	Displays the date and time of the last HUP signal. Corresponds to ndc reload .
Last Dump	Displays the date and time of the last INT signal. Corresponds to ndc dumpdb .
Total Requests	Displays the number of requests made.
Seconds Up	Displays the number of seconds elapsed since the last reboot.
Average Requests Per Second Since Start Time	Displays the average number of requests per second since the system was booted. Depending on your site's traffic, 3DNS Controller may be capable of handling a greater number of requests per second.
Average Requests Per Second Since Last Dump	Displays the average number of requests per second since the last refresh of summary statistics. Depending on your site's traffic, 3DNS Controller may be capable of handling a greater number of requests per second.

Table 7.1 General statistics (summary)

Datacenter

Item	Description
Total Datacenters	Displays the number of configured data centers in your network.
Total Servers	Displays the number of defined servers (3DNS Controllers, BIG/ip Controllers, and hosts) in your network.

Table 7.2 Datacenter statistics (summary)

BIG/ip

Item	Description
Total Servers	Displays the number of BIG/ip Controllers managed by the 3DNS Controller.
Unknown	Displays the number of BIG/ip Controllers for which the status is not known.
Up	Displays the number of BIG/ip Controllers managed by the 3DNS Controller currently marked up .
Down	Displays the number of BIG/ip Controllers managed by the 3DNS Controller currently marked down .
Waiting	Displays the number of BIG/ip Controllers managed by the 3DNS Controller currently in waiting mode.
Alert	Displays the number of BIG/ip Controllers managed by the 3DNS Controller currently in alert mode.
Panic	Displays the number of BIG/ip Controllers managed by the 3DNS Controller currently in panic mode.
Average Packet Rate	Displays the average number of packets per second in and out of the BIG/ip Controller.
Average Connections	Displays the average number of connections from the start time to the current time.
Average Nodes	Displays the number of total nodes up divided by the number of BIG/ip Controllers.

Table 7.3 BIG/ip Controller statistics (summary)

Host

Item	Description
Total Hosts	Displays the number of host machines managed by the 3DNS Controller.
Up	Displays the number of host machines managed by the 3DNS Controller currently marked up .
Down	Displays the number of host machines managed by the 3DNS Controller currently marked down .

Table 7.4 Host statistics (summary)

Virtual servers

Item	Description
Total Virtual Servers	Displays the total number of virtual servers managed by the 3DNS Controller.
Total BIG/ip Virtual Servers	Displays the number of virtual servers managed by BIG/ip Controllers.
--Up	Displays the number of BIG/ip virtual servers that are up .
--Down	Displays the number of BIG/ip virtual servers that are down .
Total Host Virtual Servers	Displays the number of virtual servers managed by a host machine.
--Up	Displays the number of host virtual servers that are up .
--Down	Displays the number of host virtual servers that are down .

Table 7.5 Virtual server statistics (summary)

Wide IP

Item	Description
Total Wide IPs	Displays the number of defined wide IPs.
Total Requests	Displays the number of name resolution requests sent to the 3DNS Controller.
Total Non-Wide IP Requests	Displays the number of regular DNS requests not intended to be load balanced.
Total Wide IP Requests	Displays the number of requests sent to a wide IP for resolution and load balancing.
Total Resolved	Displays the number of successful name resolutions.
--By Preferred	Displays the number of resolutions made using the preferred load balancing method.
--By Alternate	Displays the number of resolutions made using the alternate load balancing method.
--By Fallback	Displays the number of resolutions made using the fallback load balancing method.
Total Returned to DNS	Displays the number of name resolution requests that are returned to DNS.

Table 7.6 Wide IP statistics (summary)

Local DNS

Item	Description
Total Local DNS	Displays the number of local DNS systems accessed by the 3DNS Controller.
Probed by ICMP	Displays the number of local DNS systems accessed by the 3DNS Controller that are probed by ICMP.
Probed by TCP	Displays the number of local DNS systems accessed by the 3DNS Controller that are probed by TCP.
Probed by UDP	Displays the number of local DNS systems accessed by the 3DNS Controller that are probed by UDP.
--Needs Probe	Displays the number of local DNS systems that need to be probed.
--Idle	Displays the number of local DNS systems that were successfully probed and are waiting for the next probe.
--In Probe	Displays the number of local DNS systems that are currently being probed.
--Needs Discovery	Displays the number of local DNS systems that failed a probe.
--In Discovery	Displays the number of local DNS systems that are currently being scanned.
--Suspended	Displays the number of local DNS systems that failed the scan and are no longer eligible for probing or scanning.
Ports Discovered	Displays the number of local DNS systems whose ports have been discovered.

Table 7.7 Local DNS statistics (summary)

Path

Item	Description
Total Paths	Displays the number of paths used by the 3DNS Controller.
Paths Probed Successfully	Displays the number of paths that were successfully probed. The percentage of paths that were successfully probed is shown in parenthesis.
Fresh Paths	Displays the number of paths refreshed within the TTL. The percentage of successful probes is shown in parenthesis.
Current Average RTT	Displays the average of current RTT metrics for all paths.
Overall Average RTT	Displays the overall average round trip time for all paths. By comparing current versus overall averages, you can tell whether, on average, the current RTTs are higher or lower than the accumulative average.
Current Average Completion Rate	Displays the average of current metrics for the percentage of completed packets versus lost packets.
Overall Average Completion Rate	Displays the overall percentage of completed packets versus lost packets. By comparing current versus overall averages, you can tell whether, on average, the current completion rate is higher or lower than the accumulative average.
Paths Traced Successfully	Displays the number of paths successfully traced.
Current Average Hops	Displays the average of current metrics for the number of hops between the client and the virtual server.
Overall Average Hops	Displays the overall average of hops metrics information.
Total Picks	Displays the number of times (for all paths) where the path's data resulted in the corresponding virtual server being chosen for a connection.
Total Accesses	Displays the number of times all paths were considered when performing dynamic load balancing.
Average Outstanding Requests	Displays the number of iQuery requests made by the 3DNS Controller to a particular server that were dropped or not serviced within the timer_get_data timeframe.

Table 7.8 Path statistics (summary)

Viewing current global values

From the **Statistics** list, click **Globals** to view information about the current and default values for each **globals** sub-statement. The table also indicates whether any changes you make require that you restart **named**.

Viewing data center statistics

From the **Statistics** list, click **Data Centers** to view the following information about the data centers, and the servers they contain, in your network.

Item	Description
Refresh Interval	Specifies how often to refresh the statistical information in this table.
Data Center	Displays the name of the data center. You can click the name to link to a configuration page for the specified data center.
Name	Displays the server's name.
Server	Indicates the current status of the specified server. A green light indicates that the specified server is up ; red indicates that it is down ; blue indicates that the server's status is unknown ; yellow indicates that the server is in maintenance or standby mode and is unavailable .
Big3d	Indicates whether the big3d agent is running on the specified server. A green light indicates that the big3d agent is running; red indicates that it is not; blue indicates that the state of the big3d agent is unknown.
Type	Indicates whether the specified server is a 3DNS Controller, a BIG/ip Controller, or a host.
Address	Displays the IP address of the specified server.
BIG/ip	Displays the number of BIG/ip factories running on the specified server.
Prober	Displays the number of prober factories running on the specified server.
Discovery	Displays the number of discovery factories running on the specified server.
Hops	Displays the number of hops factories running on the specified server.
SNMP	Displays the number of SNMP factories running on the specified server.

Table 7.9 Data center statistics

Viewing sync group statistics

From the **Statistics** list, click **Sync Groups** to view the following information about the sync groups in your network.

Item	Description
Refresh Interval	Specifies how often to refresh the statistical information in this table.
Sync Group	Displays the name of the sync group. You can click the name to link to a configuration page for the specified sync group.
Named	Indicates whether named is running on the specified 3DNS Controller. A green light indicates that named is running; red indicates that it is not; blue indicates that the state of named is unknown.
Big3d	Indicates whether a big3d agent is running on the specified 3DNS Controller. A green light indicates that the big3d agent is running; red indicates that it is not; blue indicates that the state of the big3d agent is unknown.
Server Name	Displays the name of the 3DNS Controller.
Address	Displays the IP address of the 3DNS Controller.
Sync State	Indicates whether the specified 3DNS Controller is a principal or receiver.

Table 7.10 Sync group statistics

Viewing wide IP statistics

From the **Statistics** list, click **Wide IPs** to view the following information about each configured wide IP on your network. The F5 Configuration utility generates a separate row for each wide IP.

Item	Description
Refresh Interval	Specifies how often to refresh the statistical information in this table.
Domain Name	Displays the domain name for the specified wide IP. You can click the name to link to a configuration page for the specified wide IP.
TTL	Displays the TTL value specified in the wide IP definition that is passed back to the local DNS with the A record.
DNS Address	Displays the A record for the specified domain.
Service	Displays the port number or service used by the specified wide IP. If the service is a WKS (well-known service), the service name is shown. Otherwise, the port number is shown.
VSb Ratio	Displays the weighting value for the virtual servers owned by BIG/ip Controllers.
VSh Ratio	Displays the weighting value for the virtual servers owned by other host machines.
VSb LB Mode	Displays the load balancing mode in use for the pool of virtual servers owned by a BIG/ip Controller.
VSh LB Mode	Displays the load balancing mode in use for the pool of virtual servers owned by a host machine.
VSb Count	Displays the number of virtual servers owned by a BIG/ip Controller that are used to load balance the specified wide IP.
VSh Count	Displays the number of virtual servers owned by a host machine that are used to load balance the specified wide IP.
Preferred	Displays the number of times a resolution request was resolved using the preferred load balancing method specified in the wide IP definition.
Alternate	Displays the number of times a resolution request was resolved using the alternate load balancing method specified in the wide IP definition.
Fallbacks	Displays the number of times a resolution request was resolved using the fallback load balancing method specified in the wide IP definition.
Returned to DNS	Displays the number of name resolution requests that 3DNS Controller could not resolve. These requests are returned to DNS.
Last Resolution	Displays the last time this name was resolved.

Table 7.11 Wide IP statistics

Viewing 3DNS Controller statistics

From the **Statistics** list, click **3DNS** to view the following information about each 3DNS Controller in your network. The F5 Configuration utility generates a separate row for each 3DNS Controller.

Item	Description
Refresh Interval	Specifies how often to refresh the statistical information in this table.
Named	Indicates whether named is running on this 3DNS Controller. A green light indicates that named is running; red indicates that it is not; blue indicates that the state of named is unknown.
Big3d	Indicates whether the big3d agent is running on this 3DNS Controller. A green light indicates that the big3d agent is running; red indicates that it is not; blue indicates that the state of the big3d agent is unknown.
TTL	Displays the amount of time (in seconds) that the 3DNS Controller considers performance data for the other 3DNS Controllers in its sync group to be valid.
Name	Displays the name of the 3DNS Controller. You can click the name to link to a configuration page for the specified 3DNS Controller.
Address	Displays the IP address of the 3DNS Controller.
Sync Group	Displays the name of the sync group to which this 3DNS Controller belongs.
Sync Status	Indicates whether the specified 3DNS Controller is a principal or receiver.
Refreshes	Displays the number of times this data was refreshed.
Last Refresh	Displays the last time this data was refreshed.

Table 7.12 3DNS Controller statistics

Viewing BIG/ip Controller statistics

From the **Statistics** list, click **BIG/ip** to view the following information about each BIG/ip Controller in your network. The F5 Configuration utility generates a separate table for each BIG/ip Controller.

Item	Description
Refresh Interval	Specifies how often to refresh the statistical information in this table.
BIG/ip	Displays the IP address or name of the BIG/ip Controller. Click the IP address or name to link to a configuration page for the specified BIG/ip Controller.
BIG/ip OK	Displays the current status of the specified BIG/ip Controller. A green light indicates that the specified BIG/ip Controller is up ; red indicates that it is down ; blue indicates that the 3DNS Controller has not yet collected metrics from this BIG/ip Controller; yellow indicates that the BIG/ip Controller is in maintenance or standby mode and is unavailable .
Big3d OK	Indicates whether the big3d agent is up and communicating on this BIG/ip Controller. A green light indicates that the big3d agent is running; red indicates that it is not; blue indicates that the state of the big3d agent is unknown.
TTL	Displays the remaining time to live (TTL) before the BIG/ip Controller's data needs to be refreshed.
Seq No.	Displays the number of iQuery packets sent between the specified BIG/ip Controller and the 3DNS Controller.
Packets Out	Displays the number of IP packets sent by the specified BIG/ip Controller.
Packets In	Displays the total number of IP packets received by the specified BIG/ip Controller.
Packet Rate	Displays the number of packets per second in and out of the BIG/ip Controller during the last sample period.
VS Count	Displays the number of virtual servers managed by the specified BIG/ip Controller.
VS Picks	Displays the number of times a virtual server managed by the BIG/ip Controller received a resolution request from the 3DNS Controller.
Refreshes	Displays number of times this data was refreshed using the iQuery protocol.
Up Time	Displays the number of days, hours, minutes, and seconds that the specified BIG/ip Controller has been active.
Last Reply	Displays the date and time of the last contact with the specified BIG/ip Controller.

Table 7.13 BIG/ip Controller statistics

Viewing prober statistics

From the **Statistics** list, click **Probers** to view information about each configured prober in your network.

The following lines appear above the table:

❖ **Requests are sent each <number> seconds**

Indicates how often path probe requests are sent to the probers.

❖ **Regulate paths increment <number> paths**

Indicates the number of additional path probe requests to send to each prober, above the number of probe requests the prober satisfied in the previous interval. Helps ensure that each prober is working at capacity.

Item	Description
Refresh Interval	Specifies how often to refresh the statistical information in this table.
Status	Displays the current status of the specified prober. A green light indicates that the specified prober is up ; red indicates that it is down ; blue indicates that the prober's state is unknown .
Data Center	Displays the name of the data center to which this prober belongs.
Server Type	Indicates whether the prober is a 3DNS Controller or a BIG/ip Controller.
Name	Displays the name of this prober. You can click the name to link to a configuration page for the specified prober.
Prober	Displays the IP address of this prober.
Probe Cycles	Displays the number of times path data has been exchanged.
Last Sent	Displays the number of paths sent to other servers in the data center.
Avg. Sent	Displays the average number of paths sent.
Last Received	Displays the number of paths received from another server in the data center.
Avg. Received	Displays the average number of paths in a path exchange.
Next Request	Displays the number of seconds until the next exchange of path data.
# Fresh LDNS	Displays the number of local DNS entries that were updated within the TTL.
Percent of Total	Displays the ratio of fresh local DNS entries to all local DNS entries.
Get Histogram	Returns a table (in Excel format) of QOS scoring data.
Get Metrics	Returns a table (in Excel format) of QOS scoring data.

Table 7.14 Prober statistics

Viewing host statistics

From the **Statistics** list, click **Hosts** to view the following information about the generic host machines in your network. The F5 Configuration utility generates a separate row for each host. The host's IP address appears in the third column of each row; the rest of the row provides information for that host.

Item	Description
Refresh Interval	Specifies how often to refresh the statistical information in this table.
OK	Indicates the current status of the specified host machine. A green light indicates that the specified host is up ; red indicates that it is down ; blue indicates that the host is new to the 3DNS Controller and that the 3DNS Controller has not yet collected metrics from it.
TTL	Displays the remaining time to live (TTL) before a host's metrics data needs to be refreshed.
Interface Address	Displays the IP address associated with the interface that accepts incoming connections for the host. You can click the address to link to a configuration page for the specified host.
Probe Port	Displays the port that the 3DNS Controller uses to verify whether the virtual server is available.
VS Count	Displays the number of virtual servers managed by the specified host machine.
Prober	Displays the IP address of the machine owning the currently running the big3d agent process.
Protocol	Displays the protocol used for this connection.
Picks	Displays the number of times this host machine was chosen by a wide IP for load balancing.
Agent	Displays the type of SNMP agent running on the host.
Pkt Rate	Displays the current IP packet rate for SNMP monitored hosts.
Conn Rate	Displays the current connection rate for SNMP monitored hosts.
Mem Avail (Kbytes)	Displays the current amount of available memory for SNMP monitored hosts.
CPU % Used	Displays the current CPU load for SNMP monitored hosts.
Disk Avail (Kbytes)	Displays the current amount of available disk space for SNMP monitored hosts.
Refreshes	Displays the number of times this data was refreshed.
Last Refresh	Displays the last time the 3DNS Controller received data about the specified host.

Table 7.15 Host statistics

Viewing virtual server statistics

From the **Statistics** list, click **Virtual Servers** to view the following information about each configured virtual server on your network. The F5 Configuration utility generates a separate row for each virtual server.

Item	Description
Refresh Interval	Specifies how often to refresh the statistical information in this table.
OK	Indicates whether the specified virtual server is taken into consideration for load balancing. A green light indicates that the specified virtual server is up ; red indicates that it is down ; blue indicates that the virtual is unknown (new to the 3DNS Controller and that the 3DNS Controller has not yet collected metrics from it); yellow indicates that it is unavailable . See <i>Virtual server decision criteria</i> , on page 7-54.
TTL	Displays the remaining time to live (TTL) before a virtual server's metrics data needs to be refreshed.
Type	Indicates whether the specified virtual server is managed by a BIG/ip Controller (VSb) or other host machine (VSh).
Virtual Address	Displays the IP address of the specified virtual server.
Virtual Port	Displays the port number of the specified virtual server.
Ratio	Displays the weighting value for the specified virtual server.
Connections	Displays the number of current connections to the specified virtual server.
Conn Limit	Indicates whether the connection limit for this virtual server has been reached. Open indicates that the connection limit has not been reached and Full indicates that it has.
Nodes Up	Displays the number of nodes currently servicing the specified virtual server.
Enabled	Indicates whether the specified virtual server is available.
Picks	Displays the number of times this virtual server was chosen by a wide IP for load balancing.
Refreshes	Displays the number of times this data was refreshed.
Total Bytes	Displays the total number of bytes in and out on this port during the current interval.
Total Packets	Displays the total number of packets in and out on this port during the current interval.
Last Refresh	Displays the last time the 3DNS Controller received data about the specified virtual server.

Table 7.16 *Virtual server statistics*

Virtual server decision criteria

A virtual server is available to be used in a load balancing decision only if the following conditions are met:

- ❖ The BIG/ip Controller or host that manages the virtual server is available.
- ❖ The virtual server is enabled.
- ❖ The virtual server's connection limit is not exceeded.
- ❖ The number of nodes servicing the virtual server is greater than 0.
- ❖ The data was refreshed within the specified TTL. The virtual server's TTL is specified on the System - Timers & Task Intervals screen in the F5 Configuration utility, or in the **globals** sub-statement **vs_ttl**.

Viewing path statistics

From the **Statistics** list, click **Paths** to view the following path information for your network. Paths are dynamically created by the 3DNS Controller for each name resolution request. The F5 Configuration utility generates a separate row for each probe-to-local DNS path. The total number of paths is shown at the bottom of the table.

Item	Description
Refresh Interval	Specifies how often to refresh the statistical information in this table.
TTL	Displays the remaining time to live (TTL) before a path's metrics data needs to be refreshed.
Local DNS	Displays the IP address of the local DNS associated with this path.
Prober	Displays the IP address of the prober associated with this path.
RTT	Displays the average round trip time (in microseconds) for transactions between the specified prober and the local DNS.
Delta RTT	Displays the difference (in microseconds) between the current known round trip time and the average round trip time.
Hops	Displays the number of hops on the between the BIG/ip Controller and the local DNS.
Completion Rate	Displays the percentage of completed packets versus lost packets, multiplied by 100.
Picks	Displays the number of times the specified path was chosen by a wide IP for load balancing.
Accesses	Displays the number of times the specified path was evaluated to be chosen.
Last Refresh	Displays the last time this data was refreshed.

Table 7.17 Path statistics

Viewing local DNS statistics

From the **Statistics** list, click **Local DNS** to view the following information about each discovered local DNS on your network. The F5 Configuration utility generates a separate row for each local DNS.

Item	Description
Refresh Interval	Specifies how often to refresh the statistical information in this table.
Rank	Provides a measure of how often this local DNS made resolution requests. 1 indicates the local DNS that was used most often, 2 indicates the next most popular, and so on.
Local DNS	Displays the IP address of the local DNS.
3DNS Requests	Displays the number of times the 3DNS Controller received a resolution request from this local DNS.
Probe Protocol	Displays the protocol (TCP, UDP, or ICMP) used in communicating with the selected local DNS.
Port	Displays the port number used in communicating with the local DNS.
State	Provides path probing and path discovery state information. The states are listed in the next table.

Table 7.18 Local DNS statistics

Probing and discovery states

The following table lists and describes the path discovery states, which are relevant for the factories run by **big3d** agents.

State	Description
Needs Probe	Indicates that the target has never been probed or scanned.
Idle	Indicates that the target was successfully probed and is waiting for next probe.
In Probe	Indicates that the target is currently being probed.
Needs Discovery	Indicates that the target failed a probe, and now needs to be scanned.
In Discovery	Indicates that the target is currently being scanned.
Suspended	Indicates that the target failed the scan and is no longer eligible for probing or scanning.

Table 7.19 Probing and discovery states



A

Wideip.conf Syntax

Overview of the wideip.conf file

The 3DNS Controller configuration file is called **/etc/wideip.conf**. The **wideip.conf** file describes a network's data centers, servers (3DNS Controllers, BIG/ip Controllers, and hosts), and wide IPs. The **wideip.conf** file consists of two types of information: statements and comments.

Your **wideip.conf** file should include at least the following definitions.

- ❖ A **datacenter** statement. If you do not create one, the 3DNS Controller creates one for each configured server and names each as follows: **implicit--<server IP address>**
- ❖ At least one virtual server, which is defined as part of a BIG/ip or host **server** statement
- ❖ At least one **server** statement defining a 3DNS Controller
- ❖ A **wideip** statement

If a **wideip.conf** file lacks complete definitions, one of the following happens:

- ❖ If the file cannot be parsed, **named** will not start.
- ❖ If the file can be parsed, 3DNS Controllers revert to standard DNS behavior.

To open the **/etc/wideip.conf** file

On the 3DNS Maintenance menu, select **Edit 3DNS Configuration**.

WARNING

*We do not recommend opening the **wideip.conf** file in a text editor. Instead, use the Maintenance menu's **Edit 3DNS Configuration** command. This command allows you to edit and save the configuration file. This command also parses the configuration file and alerts you to any syntax errors.*

Using include files

Include files are files that contain configuration information about one aspect of your network and are listed in the main configuration file (**wideip.conf**). For example, you can have one include file that defines the BIG/ip Controllers in your network, and another include file that defines all wide IPs. Both files are listed in the **wideip.conf** file in place of the actual BIG/ip **server** and **wideip** statements.

Using include files reduces the size of the **wideip.conf** file and makes it easier to manage your configuration. In addition, include files lead to better system performance because metrics for each aspect of your network are collected and dumped only to the relevant file, instead of having all metrics collected and dumped in a single, potentially unwieldy, file.

Include files are automatically created and implemented whenever one of the following occurs:

- ❖ You configure your network setup using the F5 Configuration utility.
- ❖ You perform any type of dumping operation. By default dumping operations are **on**. To turn **off** dumping set the **global** sub-statement **timer_persist_cache** to 0.

◆ Note

When include files are generated, any comments you incorporated are deleted.

To see an example of a **wideip.conf** file that incorporates include files, see *Sample 3DNS Controller configuration file using include files*, on page A-64.

Syntax for include files

The following syntax is used when incorporating include files into a **wideip.conf** file.

```
include root_in "/var/3dns/include"
include root_out "/var/3dns/include"
include global <"file_name.conf">
include datacenter <"file_name.conf">
include sync_group <"file_name.conf">
include server <"file_name.conf">
include wideip <"file_name.conf">
include 3dscript <"file_name.conf">
include topology <"file_name.conf">
```

Figure A.1 Special syntax for include files

Definition of include statements

Parameter	Description	Default
include root_in	Specifies the name of the default directory. Enclose all file names in quotation marks.	"/var/3dns/include"
include root_out	Specifies the name of the default directory.	"/var/3dns/include"
include global	Specifies the name of the file that contains the globals statement.	"globals.conf"
include datacenter	Specifies the name of the file that contains datacenter statements.	"datacenters.conf"
include sync_group	Specifies the name of the file that contains sync_group statements.	"sync_groups.conf"
include server	Specifies the name of the file that contains server statements.	"servers.conf"
include wideip	Specifies the name of the file that contains wideip statements.	"wideips.conf"

Parameter	Description	Default
include 3dscrip	Specifies the name of the file that contains production rule configuration.	"prodrules.conf"
include topology	Specifies the name of the file that contains the topology statement.	"topology.conf"
include manifest	Specifies the name of the file that the F5 Configuration utility uses to manage any production rules generated by the utility. It is important that you do not edit this statement.	"/var/f5/www/seeit/ .prodruledb/manifest "

Statements

A top-level 3DNS Controller statement begins with a keyword and may be followed by either a value, or by a block of sub-statements enclosed in braces **{}**.

You can find an example of a complete configuration file in *Sample 3DNS Controller configuration file*, on page A-50.

The 3DNS platform supports the following top-level statements:

- ❖ **globals**
Controls global 3DNS Controller configuration options and sets defaults for other statements.
- ❖ **datacenter**
Defines the group of 3DNS Controllers, BIG/ip Controllers, and hosts that reside in a single physical location.
- ❖ **sync_group**
Defines the group of 3DNS Controllers that synchronize their configuration settings and metrics data.
- ❖ **server**
Defines a 3DNS Controller, a BIG/ip Controller, or a generic host machine.
- ❖ **wideip**
Defines a wide IP. Wide IPs map a domain name to a load balancing mode and a set of virtual servers (on BIG/ip Controllers and/or other host machines).

❖ topology

Implements and defines topology-based access control, and makes it possible for you to use the topology load balancing mode (on its own and as part of the QOS mode).

Syntax rules

Keep the following rules in mind when creating and editing statements in your **wideip.conf** file:

❖ Statement order

Statements should appear in this order:

- **globals**
- **datacenter**
- **sync_group**
- **server**
- **wideip**
- **topology**

❖ Port specification

When you define 3DNS Controllers or hosts, the port specification must follow the **probe_protocol** sub-statement.

When you define a BIG/ip Controller or virtual server (on a BIG/ip Controller or on a host), the port specification must immediately follow the address specification and can take any of the following forms:

```
address <ip_addr>:<port>
```

```
address <ip_addr>  
    port <port>
```

```
address <ip_addr>  
    service <wks>
```

In the above example, **<wks>** stands for well-known service and is a quoted string representing the name of a WKS defined in the **/etc/services** file.

❖ **cur_ values**

You may notice several **cur_** values in your **wideip.conf** file; do not edit them unless you are instructed to do so by F5 technical support. For more information, see *Understanding cur_ values*, on page A-78.

Typography in syntax examples

Certain characters are used to indicate whether a parameter is mandatory or optional, or whether you can use one parameter or another.

❖ **Mandatory parameters**

Angle brackets (< >) enclose mandatory parameters where you must type the data associated with a command.

❖ **Optional parameters**

Brackets ([ ]) enclose optional parameters.

❖ **Choice of parameters**

A vertical bar (|) between two values means that either value is acceptable.

The globals statement

The **globals** statement sets up global options to be used by the 3DNS Controller, and must appear before any other statements in the **wideip.conf** file. Each **globals** sub-statement has a default setting, and you do not need to edit the **globals** statement unless you want to change a default setting. If the 3DNS Controller does not find a **globals** statement in the configuration file, the 3DNS Controller uses a **globals** block, with each option set to its default.

If you use a **globals** sub-statement more than once, the 3DNS Controller uses the last listed value and does not generate an error message. For example, if your **globals** statement contains the following lines, the 3DNS Controller uses the value 50:

```
globals {  
    bigip_ttl 100  
    host_ttl 50  
}
```

Figure A.2 Multiple globals sub-statements

Syntax for the globals statement

The **globals** statement supports the following sub-statements.

When you define a **globals** statement, you need only include those sub-statements that you want to change from the default.

```
globals {  
  [ time_tolerance <number> ]  
  [ encryption < yes | no > ]  
  [ encryption_key_file <string> ]  
  [ check_static_depends < yes | no > ]  
  [ persist_ldns < yes | no > ]  
  [ timer_get_3dns_data <number> ]  
  [ timer_get_bigip_data <number> ]  
  [ timer_get_host_data <number> ]  
  [ timer_get_vs_data <number> ]  
  [ timer_get_path_data <number> ]  
  [ timer_get_trace_data <number> ]  
  [ timer_check_keep_alive <number> ]  
  [ timer_persist_cache <number> ]  
  [ 3dns_ttl <number> ]  
  [ bigip_ttl <number> ]  
  [ host_ttl <number> ]  
  [ vs_ttl <number> ]  
  [ path_ttl <number> ]  
  [ trace_ttl <number> ]  
  [ default_ttl <number> ]  
  [ rtt_timeout <number> ]  
  [ rtt_sample_count <number> ]  
  [ rtt_packet_length <number> ]  
  [ rtt_probe_protocol < icmp | udp | tcp > ]  
  [ rx_buf_size <number> ]  
  [ tx_buf_size <number> ]  
  [ qos_coeff_rtt <number> ]  
  [ qos_coeff_completion_rate <number> ]  
  [ qos_coeff_packet_rate <number> ]  
}
```

Figure A.3 Syntax for the globals statement (continued on next page)

```

[ qos_coeff_topology <number> ]
[ qos_coeff_hops <number> ]
[ qos_factor_rtt <number> ]
[ qos_factor_completion_rate <number> ]
[ qos_factor_packet_rate <number> ]
[ qos_factor_topology <number> ]
[ qos_factor_hops <number> ]
[ default_alternate < ga | null | random | ratio |
  return_to_dns | rr | topology > ]
[ default_fallback < completion_rate | ga | hops | leastconn |
  null | packet_rate | qos | random | ratio | return_to_dns |
  rr | rtt | topology > ]
[ fb_respect_depends < yes | no > ]
[ fb_respect_acl < yes | no > ]
[ path_hi_water <number> ]
[ path_lo_water <number> ]
[ path_duration <number> ]
[ path_reap_alg < 0 | 1 > ]
[ ldns_hi_water <number> ]
[ ldns_lo_water <number> ]
[ ldns_duration <number> ]
[ ldns_reap_alg < 0 | 1 > ]
[ prober <ip_addr> ]
[ resolver_tx_buf_size <number> ]
[ resolver_rx_buf_size <number> ]
[ use_alternate_iq_port < yes | no > ]
[ multiplex_iq < yes | no > ]
[ paths_never_die < yes | no > ]
[ paths_noclobber < yes | no > ]
[ check_dynamic_depends < yes | no > ]
[ rtt_probe_dynamic < yes | no > ]
[ rtt_port_discovery < yes | no > ]
[ rtt_autorecover_discovery < yes | no > ]
[ rtt_discovery_method < short | wks | full | all > ]
}

```

Figure A.4 Syntax for the *globals* statement (continued from previous page)

Figure A.5 shows an example a valid **globals** statement:

```
globals {
  prober 192.168.101.2      // Default prober is New York 3DNS
  encryption yes           // Encrypt iQuery
  paths_noclobber yes       // Don't overwrite metrics with
                           // zeroed results
  path_ttl 2400             // Extend the life of path metrics
  rtt_probe_dynamic yes     // Switch probing method if current
                           // fails
}
```

Figure A.5 Example syntax for the globals statement

Definition of globals sub-statements

Each **globals** sub-statement supports the parameters described below.

Synchronization

The synchronization sub-statements specify how the current 3DNS Controller handles synchronizing its database with the other 3DNS Controllers in the network.

Parameter	Description	Default
time_tolerance	Specifies the variation of time allowed (in seconds) when comparing time stamps on files. The syncd daemon allows for slight variation in time stamps when it compares files during the synchronization process. If the difference between the two time stamps falls within the time_tolerance setting, the daemon considers the files to be the same and does not overwrite one with the other. See <i>Setting the time tolerance value</i> , on page 4-29.	10

Table A.1 Synchronization sub-statements

Encryption

The encryption sub-statements specify whether the communication between the 3DNS Controller and a BIG/ip Controller is encrypted.

Parameter	Description	Default
encryption	Specifies whether to enable encryption for iQuery events.	no
encryption_key_file	Specifies the location and name of the iQuery encryption key file.	"/etc/F5key.dat"

Table A.2 Encryption sub-statements

Dependencies

The dependencies sub-statement specifies whether the 3DNS Controller checks the availability of virtual servers on BIG/ip Controllers or hosts before the 3DNS Controller sends a connection to the virtual server.

Parameter	Description	Default
check_static_depends	Specifies whether to check the availability of virtual servers on BIG/ip Controllers and hosts. Change this option to no if you want to test your configuration. Settings this option to no has the effect of forcing the virtual servers to have green (up) status indicators on the virtual servers statistics page in the F5 Configuration utility.	yes

Table A.3 Dependencies sub-statement

LDNS Persistence

The value for **persist_ldns** must be set to **yes** (the default) in order for the 3DNS Controller to store and use path information. Dynamic load balancing modes depend on path information to resolve requests. If you only use static load balancing modes, you can set **persist_ldns** to **no** to conserve memory.

Parameter	Description	Default
persist_ldns	Specifies whether the 3DNS Controller records in its cache the IP addresses of all LDNS machines that make resolution requests.	yes

Table A.4 *LDNS persistence*

Periodic task intervals

These sub-statements define the frequency at which the 3DNS Controller refreshes the metrics information it collects.

Parameter	Description	Default
timer_get_3dns_data	Specifies the frequency (in seconds) for retrieving performance data for other 3DNS Controllers in the sync group. You can enter a value between 0 and 4294967295 seconds.	20
timer_get_bigip_data	Specifies how often the 3DNS Controller refreshes BIG/ip Controller information. You can enter a value between 0 and 4294967295 seconds.	20
timer_get_host_data	Specifies how often the 3DNS Controller refreshes other host machine information. You can enter a value between 0 and 4294967295 seconds.	90
timer_get_vs_data	Specifies how often the 3DNS Controller refreshes virtual server information. You can enter a value between 0 and 4294967295 seconds.	30
timer_get_path_data	Specifies how often the 3DNS Controller refreshes path information (for example, round trip time or ping packet completion rate). You can enter a value between 0 and 4294967295 seconds.	120
timer_get_trace_data	Specifies the frequency (in seconds) for retrieving trace route data (trace routes between each data center and each local DNS). You can enter a value between 0 and 4294967295 seconds.	60
timer_check_keep_alive	Specifies how often the 3DNS Controller queries remote 3DNS Controllers and BIG/ip Controllers. This value determines how often named sends hello packets to each big3d agent in its configuration. You can enter a value between 0 and 4294967295 seconds.	60
timer_persist_cache	Specifies the frequency (in seconds) that the 3DNS Controller writes the wideip.conf file from memory. You can enter a value between 0 and 4294967295 seconds.	300

Table A.5 Periodic task interval statements

Data timeouts

These sub-statements set the amount of time for which metrics information is considered valid. After a timeout is reached, the 3DNS Controller refreshes the information.

Parameter	Description	Default
default_ttl	Specifies the default A record TTL value for wide IPs. If you do not specify a TTL value in a wide IP definition, it takes on the value of default_ttl .	30
3dns_ttl	Specifies the amount of time (in seconds) that the 3DNS Controller considers performance data for the other 3DNS Controllers to be valid.	60
bigip_ttl	Specifies the amount of time (in seconds) that BIG/ip Controller information is to be used by the 3DNS Controller for name resolution and load balancing. You can enter a value between 1 and 4294967295. The following relationship should be maintained: bigip_ttl > timer_get_bigip_data . A 2:1 ratio is the optimal setting for this relationship.	60
host_ttl	Specifies the amount of time (in seconds) that generic host machine information is to be used by the 3DNS Controller for name resolution and load balancing. You can enter a value between 1 and 4294967295. The following relationship should be maintained: host_ttl > timer_get_host_data .	240
vs_ttl	Specifies the amount of time (in seconds) that virtual server information (data acquired from a BIG/ip Controller or other host machine about a virtual server) is to be used by the 3DNS Controller for name resolution and load balancing. You can enter a value between 1 and 4294967295. The following relationship should be maintained: vs_ttl > timer_get_vs_data .	120
path_ttl	Specifies the amount of time (in seconds) that path information is to be used by the 3DNS Controller for name resolution and load balancing. You can enter a value between 1 and 4294967295. The following relationship should be maintained: path_ttl > timer_get_vs_data .	600
trace_ttl	Specifies the amount of time (in seconds) that the 3DNS Controller considers trace route data to be valid. You can enter a value between 1 and 4294967295.	604800 (seven days)

Table A.6 Data timeouts sub-statements

Metrics collection

The metrics collection sub-statements define how the 3DNS Controller collects path information.

Parameter	Description	Default
rtt_timeout	Specifies how long the big3d agent waits for a probe. You can enter a value between 1 and 4294967295 seconds.	5
rtt_sample_count	To determine path information between a local DNS and a BIG/ip Controller, the number of packets (specified by rtt_sample_count) of certain length (specified by rtt_packet_length) is sent via ping from the BIG/ip Controller to the local DNS. You can enter a value between 1 and 25.	3
rtt_packet_length	To determine path information between a local DNS and a BIG/ip Controller, the number of packets (specified by rtt_sample_count) of certain length (specified by rtt_packet_length) is sent via ping from the BIG/ip Controller to the local DNS. You can enter a value between 64 and 500.	64
rtt_probe_protocol	Specifies a probe method to calculate RTT times. You can specify the ICMP, UDP, or TCP protocol.	icmp

Table A.7 Metrics collection sub-statements

Resource limits

The resource limits sub-statements define the amount of memory allocated to sending and receiving metrics information.

Parameter	Description	Default
rx_buf_size	Specifies the maximum amount of socket buffer data memory the server can use when receiving data. You can enter a value between 8192 and 65536.	49152
tx_buf_size	Specifies the maximum amount of socket buffer data memory the server can use when transmitting data. You can enter a value between 8192 and 65536.	49152

Table A.8 Resource limits sub-statements

QOS values

The Quality of Service (QOS) load balancing mode distributes connections based on a path evaluation score. Using the equation below, the QOS mode compares paths between the local DNS and each virtual server included in the **wideip** statement. The 3DNS Controller load balances each new connection to the virtual server associated with the best (highest) path score.

```
score_path =
[(qos_coeff_packet_rate) * (1 / score_packet_rate)] +
(qos_coeff_rtt) * (1 / score_rtt)] +
[(qos_coeff_completion_rate) * (score_completion_rate)] +
[(qos_coeff_topology) * (score_topology)] +
[(qos_coeff_hops) * (score_hops)]
```

Figure A.6 QOS equation

The coefficients for the score computation are defined as **globals**, but you can override them within a **wideip** statement.

Parameter	Description	Default
qos_coeff_rtt	Specifies the relative weighting for round trip time when the load balancing mode is set to Quality of Service. You can enter a value between 0 and 100.	20
qos_coeff_completion_rate	Specifies the relative weighting for ping packet completion rate when the load balancing mode is set to Quality of Service. You can enter a value between 0 and 100.	5
qos_coeff_packet_rate	Specifies the relative weighting for BIG/ip Controller packet rate when the load balancing mode is set to Quality of Service. You can enter a value between 0 and 100.	3
qos_coeff_topology	Specifies the relative weighting for topology when the load balancing mode is set to Quality of Service. You can enter a value between 0 and 100.	0
qos_coeff_hops	Specifies the relative weighting for hops when the load balancing mode is set to Quality of Service. You can enter a value between 0 and 100.	0
qos_factor_rtt	Specifies the factor used to normalize raw RTT values when computing the QOS score.	10000
qos_factor_completion_rate	Specifies the factor used to normalize raw completion rate values when computing the QOS score.	10000
qos_factor_packet_rate	Specifies the factor used to normalize raw packet rate values when computing the QOS score.	10000
qos_factor_topology	Specifies the factor used to normalize raw topology values when computing the QOS score.	10
qos_factor_hops	Specifies the factor used to normalize raw hops values when computing the QOS score.	25

Table A.9 QOS values sub-statements

Load balancing

Parameter	Description	Default
default_alterate	Defines the default load balancing mode used for the alternate method (formerly default_static). You can override this setting in the wideip statement.	rr
default_fallback	Defines the default load balancing mode used for the fall-back method. You can override this setting in the wideip statement.	return_to_dns
fb_respect_depends	Determines whether the 3DNS Controller respects virtual server status when load balancing switches to the specified fallback mode.	no
fb_respect_acl	Determines whether the 3DNS Controller imposes topology access control when load balancing switches to the specified fallback mode.	no

Table A.10 Load balancing sub-statements

Prober

The **prober** sub-statement defines the IP address of the machine that pings a host system to verify whether it is available. Typically, you use the IP address of the 3DNS Controller itself, but you can use other network servers.

Parameter	Description	Default
prober	Specifies the default prober for host status, usually the 3DNS Controller IP address. Using this sub-statement is not necessary if the 3DNS Controller only manages the BIG/ip Controller. When this option is set to 0, the 3DNS Controller's IP address is the implied value. This sub-statement can be overridden within the server statement.	0.0.0.0

Table A.11 *Prober sub-statement*

 WARNING

You must define a prober if the 3DNS Controller manages virtual servers on hosts. If you do not define a default prober in the globals, or probers in the host statements for all hosts, you will encounter validation errors.

Buffer size

The buffer size sub-statements specify the maximum amount of UDP data that the 3DNS Controller can receive, and also specify the maximum amount of TCP data that the 3DNS Controller can send.

Parameter	Description	Default
resolver_rx_buf_size	Specifies the UDP receive buffer size. The value is overridden only if it is larger than the one first assigned by the kernel.	98304
resolver_tx_buf_size	Specifies the TCP send buffer size.	24576

Table A.12 *Buffer size sub-statements*

Reaping

The 3DNS Controller stores dynamic LDNS and network path data in memory. The amount of data that can be held in memory at any given time is based on the amount of memory in the 3DNS Controller. Reaping is the process of finding the least-used data in memory and deleting it.

The default reaping values are adequate for most configurations. Contact F5 technical support if you want to make changes to them.

◆ Note

*The default values for **path_hi_water**, **path_lo_water**, **ldns_hi_water**, and **ldns_lo_water** vary depending on available memory and are automatically established during the startup process.*

Parameter	Description	Default
path_hi_water	Specifies the high water mark, in bytes, for reaping paths.	varies
path_lo_water	Specifies the low water mark, in bytes, for reaping paths.	varies
path_duration	Specifies how often an event is triggered that calls the reaping function for paths. You can enter a value between 3600 and 2419200 seconds.	345600 (4 days)
path_reap_alg	Specifies the method by which unexpired paths are reaped during the general reap process. You can enter 0, which corresponds to least recently used, or 1, which corresponds to least number of hits.	0
ldns_hi_water	Specifies the high water mark, in bytes, for reaping LDNS systems.	varies
ldns_lo_water	Specifies the low water mark, in bytes, for reaping LDNS systems.	varies
ldns_duration	Specifies how often an event is triggered that calls the reaping function for LDNS systems. You can enter a value between 3600 and 2419200.	2419200 (28 days)
ldns_reap_alg	Specifies the method by which unexpired LDNS systems are reaped during the general reap process. The only valid value is 0, which corresponds to least recently used.	0

Table A.13 Reaping sub-statements

iQuery port options

Parameter	Description	Default
use_alterdate_iq_port	Determines whether the 3DNS Controller runs iQuery traffic on port 245 (the port used in older configurations), or on the IANA registered iQuery port, 4353. The default setting, no , uses port 245. To use port 4353, change this setting to yes .	no
multiplex_iq	Determines whether the 3DNS Controller uses the ephemeral ports for iQuery traffic returned from the big3d agent. To force iQuery traffic to use a single port defined by use_alterdate_iq_port for all incoming iQuery traffic, change this setting to yes .	no

Table A.14 iQuery port options sub-statements

Probing

Parameter	Description	Default
paths_never_die	Specifies that dynamic load balancing modes can use path data even after the TTL for the path data has expired. We recommend that you change this setting to yes , which has the effect of requiring that the 3DNS Controller always uses path data even if the path's TTL expires.	no
paths_noclobber	Specifies whether the 3DNS Controller overwrites existing path data with blank data when a path probe fails. We recommend that you change this setting to yes , which has the effect of requiring that the 3DNS Controller does not overwrite existing path data with blank data when a path probe fails. Unlike paths_never_die , this parameter has no effect on path_ttl .	no
check_dynamic_depends	Specifies that the 3DNS Controller checks the availability of a path before it uses the path for load balancing. Changing this option to no overrides the path_ttl and whether the last probe attempt was successful. You can use this parameter in conjunction with paths_noclobber . This parameter does not prevent the refreshing of path metrics.	yes
rtt_probe_dynamic	Determines whether the 3DNS Controller attempts additional probes using alternate probe protocols if the probe protocol specified by rtt_probe_protocol fails during the first probe.	no

Parameter	Description	Default
rtt_port_discovery	Determines whether the 3DNS Controller uses the discovery factory to find an alternate port to be used by the probing factory, if probing on port 53 fails.	no
rtt_autorecover_discovery	Specifies whether to move collected LDNS information from the Needs Discovery probing state to the Needs Probe state if rtt_port_discovery is set to no . Setting this to no means that if probing failed for a specific LDNS for any reason, the LDNS is ineligible for future probing attempts.	yes
rtt_discovery_method	Determines which ports to scan. The default, short , causes the 3DNS Controller to scan a pre-defined list of ports, and then scans port 53. Other acceptable values are wks (well-known services), full (all ports between 1 and 1024), and all .	short

Table A.15 Probing sub-statements

The datacenter statement

A **datacenter** statement defines the group of 3DNS Controllers, BIG/ip Controllers, and hosts that reside in a single physical location.

Syntax for the datacenter statement

The **datacenter** statement uses the following syntax.

```
datacenter {
  name <"data center name">
  [ location <"location info"> ]
  [ contact <"contact info"> ]
  [ 3dns <3DNS IP address> ]
  [ bigip <BIG/ip IP address> ]
  [ host <host IP address> ]
}
```

Figure A.7 Syntax for the datacenter statement

Figure A.8 shows an example of a valid **datacenter** statement:

```
datacenter {
  name "New York"
  location "NYC"
  contact "3DNS_Admin"
  3dns 192.168.101.2
  bigip 192.168.101.40
  host 192.168.105.40
}
```

Figure A.8 Example syntax for the datacenter statement

Definition of datacenter sub-statements

The **datacenter** sub-statements specify a name for the data center and the machines it contains.

Parameter	Description
name	Specifies the name of this data center. The name must be enclosed in quotation marks.
location	Specifies the location of the data center. This name must be enclosed in quotation marks. This sub-statement is not required, but this information can be useful if problems later arise or changes are required.
contact	Identifies the administrator of the data center. This name must be enclosed in quotation marks. This sub-statement is not required, but this information can be useful if problems later arise or changes are required.
3dns	Specifies the IP address of a 3DNS Controller in this data center.
bigip	Specifies the IP address of a BIG/ip Controller in this data center.
host	Specifies the IP address of a host in this data center.

Table A.16 Datacenter sub-statements

The sync_group statement

The **sync_group** statement defines the group of 3DNS Controllers that synchronize their configuration settings and metrics data. You configure this statement in the **wideip.conf** file of the principal 3DNS Controller.

Syntax for the sync_group statement

The **sync_group** statement uses the following syntax.

```
sync_group {  
    name <"name">  
    3dns <ip_address | "domain_name">  
    [ 3dns <ip_address | "domain_name"> ]  
}
```

Figure A.9 Syntax for the sync_group statement

Note that the **sync_group** statement does not support location or contact sub-statements.

Figure A.10 shows an example of a valid **sync_group** statement:

```
sync_group {  
    name "sync"  
    3dns 192.168.101.2    // New York  
    3dns 192.168.102.2    // Los Angeles  
}
```

Figure A.10 Example syntax for the sync_group statement

Definition of sync_group sub-statements

Parameter	Description
name	Specifies the name of this sync group.
3dns	Specifies the IP address or domain name (enclosed in quotation marks) of a 3DNS Controller in the group. First list the IP address of the principal itself. Then list all other 3DNS Controllers, in the order that they should become principals should previously listed 3DNS Controllers fail.

Table A.17 Sync_group sub-statements

The server statement

The **server** statement defines the characteristics associated with a particular 3DNS Controller, BIG/ip Controller, or host.

◆ **Note**

*The **server** statement replaces the **bigip** and **host** statements used in earlier versions of 3DNS Controller. Although this version of 3DNS Controller provides backward compatibility with the earlier **bigip** and **host** statements, we recommend that you use the newer syntax.*

A **server** statement contains the following information:

- ❖ The type of server: 3DNS Controller, BIG/ip Controller, or host
- ❖ The IP address of the server
- ❖ If the server is a BIG/ip Controller or host, the set of virtual servers that are available on it
- ❖ If the server is a BIG/ip Controller, dynamically collected information about the BIG/ip Controller, its virtual servers and ports, and the paths between the BIG/ip Controller and local DNS

Because available sub-statements vary by server type, the syntax and examples for each type are listed separately. All sub-statements are defined in the table starting on page A -34.

Syntax for the server statement (3DNS Controller)

The following **server** statement syntax applies to 3DNS Controllers only. Note that this **server** statement does not define virtual servers; the purpose of defining a 3DNS Controller is to set up the **big3d** agent to obtain path probing information.

```
server {  
    type 3dns  
    address <IP address>  
    name <"3dns_name">  
    [ remote {  
        secure <yes | no>  
        user <"user name">  
    } ]  
    [ interface {  
        address <NIC IP address>  
        address <NIC IP address>  
    } ]  
    [ factories {  
        prober <number>  
        discovery <number>  
        snmp <number>  
        hops <number>  
    } ]  
    [ prober <IP address> ]  
    probe_protocol < icmp | udp | tcp >  
    port <port to probe>  
}
```

Figure A.11 Syntax for defining a 3DNS Controller

Figure A.12 shows an example of the syntax to use in defining a 3DNS Controller:

```
// New York
server {
    type 3dns
    address 192.168.101.2
    name "3dns-newyork"
    remote {
        secure no
        user "root"
    }
    probe_protocol icmp
    port 53
}
```

Figure A.12 Example syntax for defining a 3DNS Controller

Syntax for the server statement (BIG/ip Controller)

The following **server** statement syntax applies to BIG/ip Controllers only.

```
server {
  type bigip
  address <IP address>
  name <"bigip_name">
  [ remote {
    secure <yes | no>
    user <"user name">
  } ]
  [ interface {
    address <NIC IP address>
    address <NIC IP address>
  } ]
  [ factories {
    prober <number>
    discovery <number>
    snmp <number>
    hops <number>
  } ]

  vs {
    address <virtual server IP address>
    port <port number> | service <"service name">
    [ translate {
      address <IP address>
      port <port number>|service <"service name">
    } ]
  }
}
```

Figure A.13 Syntax for defining a BIG/ip Controller

Figure A.14 shows an example of the syntax to use in defining a BIG/ip Controller:

```
server {
    type          bigip
    address       192.168.101.40
    name          "bigip-newyork"

    remote {
        secure    yes
        user       "administrator"
    }

    # Tell 3DNS about the 2 interfaces on a BIG/ip HA
    interface {
        address    192.168.101.41
        address    192.168.101.42
    }

    # Change the number of factories doing the work at big3d
    factories {
        prober     6
        discovery  1
        snmp       1
        hops       2
    }

    vs {
        address    192.168.101.50
        service     "http"
        translate {
            address  10.0.0.50
            port     80
        }
    }
    vs {
        address    192.168.101.50:25 // smtp
        translate {
            address  10.0.0.50:25
        }
    }
}
```

Figure A.14 Example syntax for defining a BIG/ip Controller

Syntax for the server statement (host)

The following **server** statement syntax applies to hosts only. Note that the **snmp** sub-statement is only necessary if you want the **big3d** agent to use an SNMP agent on the host to collect additional metrics information. For more information, see *Configuring host SNMP settings*, on page 4-17.

```
server {
  type host
  address <IP address>
  name <"host_name">
  probe_protocol <tcp | icmp | udp>
  [ prober <IP address> ]
  port <port number> | service <"service name">
  [ snmp {
    agent <generic | ucd | solstice | ntserve>
    port <port number>
    community <"community string">
    timeout <seconds>
    retries <number>
    version <SNMP version>
  } ]
  vs {
    address <virtual server IP address>
    port <port number> | service <"service name">
    [ probe_protocol <tcp | icmp | udp> ]
  }
}
```

Figure A.15 Syntax for defining a host

Figure A.16 shows an example of the syntax to use in defining a host:

```
server {
  type          host
  address       192.168.104.40
  name          "host-tokyo"
  probe_protocol icmp
  port         53
  snmp {
    agent       ucd
    community   "public"
    version     1
  }
  vs {
    address     192.168.104.50:25
  }
  vs {
    address     192.168.104.50:80
  }
}
```

Figure A.16 Example syntax for defining a host

Definition of server sub-statements

The **server** statement supports the following sub-statements. Note that available sub-statements vary by server type.

Address information

The address information sub-statements specify the name, address, and type of each server. Depending on the type of server you are configuring, you may need to specify a probe protocol, prober IP address, and port number.

Parameter	Description
type	Indicates whether the specified server is a 3DNS Controller, BIG/ip Controller, or host.
address	Specifies the IP address of the 3DNS Controller, BIG/ip Controller, or host.
name	Specifies the name of the 3DNS Controller, BIG/ip Controller, or host. You must enclose all names in quotation marks.
probe_protocol	Specifies the protocol method to use for probing this host: ICMP, UDP, or TCP. Applies to 3DNS Controllers and hosts. Note that UDP is not supported on hosts.
prober	Specifies the IP address of the machine probing the 3DNS Controller or host. This IP address points to either a BIG/ip Controller or a 3DNS Controller that runs the big3d agent. The big3d agent actually probes the host and virtual servers to verify whether the host or a particular virtual server is currently available to accept connections. If you omit this parameter, the 3DNS Controller uses the prober <ip_addr> parameter defined in the globals statement. Applies to 3DNS Controller and hosts.
port	Specifies the port used to probe this host if probe_protocol is set to TCP. Applies to 3DNS Controllers and hosts.

Table A.18 Address information sub-statements

Remote connections

Using the **remote** sub-statement is only necessary if you want to specify a different log-on name or specifically use SSH or RSH on 3DNS Controllers and BIG/ip Controllers.

Parameter	Description
remote	Indicates the start of a remote sub-statement. Applies to 3DNS Controllers and BIG/ip Controllers.
secure	Specifies whether to use ssh (secure shell) or rsh (remote shell) for remote connections. The default for US controllers is yes , which specifies that ssh is used. International versions must use rsh instead. Applies to 3DNS Controllers and BIG/ip Controllers.
user	Specifies the "superuser" name that is used to allow a remote user to log on to the controller. Enclose this name in quotation marks. If you omit this parameter, the default, " root ", is used. Applies to 3DNS Controllers and BIG/ip Controllers.

Table A.19 Remote connections sub-statements

Hardware redundancy

If you have hardware-redundant 3DNS Controllers and BIG/ip Controllers, you must configure the **interface** sub-statement for the 3DNS Controller to work properly with BIG/ip Controllers in Active-Active mode. This sub-statement is also required in using the standby BIG/ip Controller or 3DNS Controller for probing.

Parameter	Description
interface	Indicates the start of the interface sub-statement.
address	Specifies the IP address of both network interface cards, on separate lines. Applies to 3DNS Controllers and BIG/ip Controllers.

Table A.20 Hardware redundancy sub-statements

Factories

With 3DNS Controllers and BIG/ip Controllers, you can change the number and types of probing factories by using the **factories** sub-statement. If you omit this sub-statement, the defaults are used. For more information on probing, see *Setting up data collection with the big3d agent*, on page 2-9.

Parameter	Description
factories	Indicates the start of the factories definition. Applies to 3DNS Controllers and BIG/ip Controllers.
prober	Specifies the number of prober factories to use.
discovery	Specifies the number of discovery factories to use.
snmp	Specifies the number of snmp factories to use.
hops	Specifies the number of hops factories to use.

Table A.21 *Factories sub-statements*

SNMP settings

The **snmp** sub-statement is valid for hosts only. This sub-statement instructs the **big3d** agent to use an SNMP agent on the host to collect additional metrics information.

If you need help configuring the SNMP agent on the host itself, see *Configuring SNMP agents on hosts*, on page 4-20.

Parameter	Description
snmp	Specifies the start of an SNMP definition. Applies to hosts only.
agent	Specifies the SNMP agent type. If you omit this parameter, the big3d agent uses the generic SNMP agent. Applies to hosts only.
port	Specifies the port the SNMP agent runs on. Applies to hosts only.
community	Specifies the password for basic SNMP security and for grouping SNMP hosts. Enclose this string in quotation marks. Applies to hosts only.
timeout	Specifies the amount of time (in seconds) for the timeout. The default is appropriate in most cases. If you are contacting a host through a very slow network, you can try increasing the timeout and retries values to improve performance. However, the problem with increasing these values is that a host that is down can hang up the SNMP for an excessive amount of time. Applies to hosts only.
retries	Specifies the number of times requests should be retried. The default is appropriate in most cases. If you are contacting a host through a very slow network, you can try increasing the timeout and retries values to improve performance. However, the problem with increasing these values is that a host that is down can hang up the SNMP for an excessive amount of time. Applies to hosts only.
version	Specifies the SNMP agent version number. Applies to hosts only.

Table A.22 *SNMP sub-statements*

Virtual server definitions

Part of defining a BIG/ip Controller server or host server is defining the virtual servers that the server manages. After you define a virtual server here (including specifying the address and port), you can use this virtual server in a **wideip** definition.

Parameter	Description
vs	Indicates the start of a virtual server definition. Applies to BIG/ip Controllers and hosts.
address	Specifies the IP address of a virtual server owned by a BIG/ip Controller or host. Note that the virtual server's address must be listed first, before port or service values. Applies to BIG/ip Controllers and hosts.
port or service	Specifies the virtual server's port number or service name. You can add the port number, preceded by a colon, on the same line as the virtual server's address, or you can enter it on the next line. You can use the service name if it is a WKS (well known service) and you enclose it in quotation marks. Applies to BIG/ip Controllers and hosts.
translate	Specifies that iQuery packets sent to the BIG/ip Controller include translated IP addresses (required if the packets must pass through a firewall). When you use this keyword, you must then include address and port/service information for the translated IP addresses. Applies to BIG/ip Controllers only.

Table A.23 Virtual server definitions

The wide IP statement

The **wideip** statement defines a wide IP. A wide IP maps a domain name to a load balancing mode and a set of virtual servers (on BIG/ip Controllers and/or other host machines).

Syntax for the wideip statement

The **wideip** statement uses the following syntax.

```
wideip {
  address <ip_addr>
  port <port_number> | <"service name">
  name <"domain_name">
  [ alias <"alias_name"> ]
  [ ttl <number> ]
  [ port_list <port_number> <port_number> ... ]
  [ qos_coeff {
    rtt <n>
    completion_rate <n>
    packet_rate <n>
    topology <n>
    hops <n>
  } ]
  [ pool_lbmode <rr | ratio | ga | random> ]
  pool {
    name <"pool_name">
    [ ratio <pool_ratio> ]
    [ dynamic_ratio < yes | no > ]
    [ rr_ldns < yes | no > ]
    [ rr_ldns_limit <number> ]
    [ preferred < completion_rate | ga | hops | leastconn |
      packet_rate | qos | random | ratio | return_to_dns | rr |
      rtt | topology > ]
    [ alternate < ga | null | random | ratio | return_to_dns |
      rr | topology | packet_rate | leastconn > ]
    [ fallback < completion_rate | ga | hops | leastconn | null |
      packet_rate | qos | random | ratio | return_to_dns | rr |
      rtt | topology > ]
    address <vs_addr>[:<port>] [ratio <weight>]
  }
}
```

Figure A.17 Syntax for the wideip statement

FigureA.18 shows an example of a valid **wideip** statement:

```
wideip {
  address      192.168.102.50
  service      "smtp"
  name         "mx.wip.domain.com"
  alias        "mail.wip.domain.com"
  pool_lbmode   ratio
  pool {
    name        "pool_1"
    ratio       3
    preferred    rtt
    alternate    random
    address      192.168.101.50
    address      192.168.102.50
    address      192.168.103.50
  }
  pool {
    name        "pool_2"
    ratio       1
    preferred    ratio
    address      192.168.104.50    ratio 2
    address      192.168.105.50    ratio 1
  }
}
```

Figure A.18 Example syntax for the wideip statement

Definition of wideip sub-statements

Wide IP sub-statements define groups virtual servers to be load balanced, and they assign load balancing characteristics, such as the load balancing mode, to each group.

Address information

The address information sub-statements specify the wide IP key. They also specify the pool of virtual servers that the wide IP load balances.

Parameter	Description
address	Specifies a key that represents one valid virtual server IP address from the set that services this wide IP. This key is also listed as the A record in the zone file for the domain.
port or service	Specifies the virtual server's default port number or service name. You can use the service name if it is a WKS (well known service) and you enclose it in quotation marks.
name	Specifies the domain name for this wide IP (for example, " www.wip.domain.com "). You must enclose all names in quotation marks.
alias	Specifies an alternate name for this wide IP. You must enclose all names in quotation marks. Alias names are treated the same as the domain name. You can specify up to 200 alias names for each wide IP.
ttl	Specifies the amount of time (in seconds) that the A record is used by the local DNS after resolving the wide IP. This is the TTL associated with the A record as specified by RFC 1035, on pages 29 and 30.
port_list	Specifies a list of ports that must be available before the 3DNS Controller can send connections to the specified address.
qos_coeff	Specifies the relative weighting for each load balancing method in calculating the Quality of Service (QOS) mode. Before adjusting QOS coefficients, see <i>Understanding QOS coefficients</i> , on page 6-2.
pool_lbmode	Specifies the load balancing mode to use to balance requests over all pools.
pool	Indicates the start of the pool definition for this wide IP. A pool is a set of virtual servers defined and owned by a BIG/ip Controller or other host machine.
name	As part of a pool definition, defines the name of this pool. All names must be enclosed in quotation marks.
ratio	As part of a pool definition, ratio specifies the default weighting to use with respect to other pool types when the pool lbmode is ratio.
dynamic_ratio	Specifies whether the 3DNS Controller treats QOS scores as ratios, and uses each server in proportion to the ratio determined by the QOS calculation. The default is no .

Parameter	Description
rr_ldns	Specifies whether the 3DNS Controller returns a list of available virtual servers available for load balancing to a client and stores the list in the browser cache. The default is no , which specifies that the 3DNS Controller returns only one A record per query.
rr_ldns_limit	The maximum number of A records to return when rr_ldns is set to yes. You can enter a value between 0 and 16. The default is 0 , which specifies that the 3DNS Controller returns the IP addresses of all (up to 16) available virtual servers.
preferred	Specifies the load balancing mode to use for the specified pool. Each acceptable value is described in the next table. The default is rr .
alternate	Specifies the load balancing mode to use for the specified pool if the preferred mode fails. The default is rr . Also see the description of default_alterate , a globals sub-statement, on page A -19.
fallback	Specifies the load balancing mode to use for the specified pool if the alternate mode fails. If the fallback mode fails, the 3DNS Controller returns the request to DNS. The default is return_to_dns . Also see the description of default_fallback , a globals sub-statement, on page A -19.
address	As part of a pool definition, address specifies the IP address of each virtual server in this pool. You can use the same virtual server in multiple pools, but not within the same pool.
port	Specifies a specific port to use for the specified virtual server. This sub-statement is optional. A port specified here overrides the wide IP's port setting. If a port is not specified here, the wide IP's port value is assumed.
ratio	As part of a virtual server's address specification, ratio defines the default weighting to use with respect to all virtual servers in this pool when the ratio load balancing mode is employed. The default is 1 .

Table A.24 Address information sub-statements

Load balancing modes

The load balancing sub-statements specify the load balancing modes to use for the wide IP in this order:

- ❖ The 3DNS Controller attempts to load balance requests using the **preferred** mode.

- ❖ If the **preferred** mode fails, the 3DNS Controller tries the **alternate** mode.
- ❖ If the **alternate** mode fails, the 3DNS Controller tries the **fallback** mode.
- ❖ If the **fallback** mode fails, the request is returned to DNS. DNS attempts to resolve the request based on the contents of the zone files.

As noted in the table below, not all modes are valid for the **alternate** sub-statement. Also note that the **alternate** and **fallback** sub-statements accept two additional values, **return_to_dns** and **null**.

If you do not specify a load balancing mode, the wide IP uses the default load balancing mode defined in the **globals** statement (see page A -6).

Parameter	Description
completion_rate	Sends each new connection to the server that has the fewest number of dropped packets. Valid in a preferred or fallback sub-statement.
global_availability (ga)	Distributes connections to a list of servers, always sending a connection to the first available server in the list.
hops	Sends each new connection to the server that has the fewest number of network hops between the server and the client LDNS. Valid in a preferred or fallback sub-statement.
leastconn	Sends each new connection to the server that currently hosts the fewest current connections.
null	Bypasses the current load balancing method and forces the 3DNS Controller to use the next load balancing method or, if it has cycled through all load balancing sub-statements for the pool, to the next pool. Valid in an alternate or fallback sub-statement.
packet_rate	Sends each new connection to the server that is managed by a BIG/ip Controller currently handling the least amount of network traffic (determined by the fewest number of packets currently processed by the controller).
qos	Takes these performance factors into account when determining how to distribute connections: hops, packet rate, completion rate, round trip time, and topology. You can configure how much emphasis to place on each performance factor, or you can configure the QOS mode to treat all factors as being equally important. Valid in a preferred or fallback sub-statement.
random	Distributes each new connection to a server chosen at random from the wide IP set of virtual servers.
ratio	Distributes new connections across servers in proportion to a user-defined ratio.
return_to_dns	Returns the resolution request to DNS, preventing the 3DNS Controller from using the next load balancing method or using the next available pool.

Parameter	Description
rr	Distributes connections evenly across all servers, passing each new connection to the next server in line.
rtt	Sends each new connection to the server that demonstrates the fastest round trip time between the server and the client LDNS. Valid in a preferred or fallback sub-statement.
topology	Distributes connections based on the proximity of a local DNS to a particular data center.

Table A.25 Load balancing sub-statements

Use the following equation to configure the QOS load balancing mode:

A (1/packet rate) + B (1/rtt) + C (completion rate) + D (topology) + E (1/hops)

The topology statement

The **topology** statement implements a form of wide-area IP filtering. Topology-based access control allows you to specify which data centers are acceptable for a given resolution request, based on the proximity of the data center's IP address to the requesting IP address of the local DNS server. For example, you can specify that requesting local DNS clients in North America are allowed access to data centers in North America, but not allowed access to data centers in South America.

By including a **topology** statement in your **wideip.conf** file, you can also use the topology load balancing mode, both on its own and as part of the QOS mode.

For more information and an example of a **topology** statement, see *Setting up Topology-based access control*, on page 6-13.

Syntax for the topology statement

The **topology** statement uses the following syntax.

```
topology {  
  acl_threshold    <0..4294967295>  
  probe_threshold  <0..4294967295>  
  limit_probes     <yes |no>  
  longest_match    <yes | no>  
                  <server cidr> <LDNS cidr> <score>  
}
```

Figure A.19 Syntax for the topology statement

Definition of topology sub-statements

Parameter	Description
acl_threshold	Specifies a score threshold. Any list record (prober IP address/local DNS IP address) score that is less than the threshold value will not have access to the listed virtual servers. You can enter a value between 0 and 4294967295.
probe_threshold	Specifies a threshold for probing. Any list record (prober IP address/local DNS IP address) score that is less than the probe threshold value prevents the local DNS from being probed by the specific prober. You can enter a value between 1 and 4294967295.
limit_probes	Specifies whether to apply access control to the probing of paths. If this parameter is set to yes , the 3DNS Controller requests a given BIG/ip Controller to probe only those local DNS servers that can connect to it according to the probe_threshold value and the topology map scores. The default is yes .
longest_match	In cases where there are several IP/mask items that match a particular IP address, longest_match specifies whether the 3DNS Controller selects the record that is most specific, and thus has the longest mask. The default is yes .
server CIDR	Specifies the server mask for a given data center. This is one of two values used to determine the longest match.
LDNS CIDR	Specifies the local DNS mask. This is one of two values used to determine the longest match.
score	Specifies the list record score, which is used for the comparison of virtual servers when the topology load balancing mode is employed.

Table A.26 Topology sub-statements

Comments

You can insert comments anywhere you would otherwise see white space in the 3DNS Controller configuration file.

Syntax

Note that the comment syntax depends on the environment in which you use the configuration file.

For example:

```
/* This is a 3DNS comment as in C */  
// This is a 3DNS comment as in C++  
# This is a 3DNS comment as in common Unix shells and Perl
```

Figure A.20 Comment syntax

Definition and usage

The format for comments varies by programming language; each format is described below. To avoid comment nesting problems, we recommend that you use only one comment style in your **wideip.conf** file. However, all styles may be used in a single **wideip.conf** file.

C style comments

C style comments start with the slash character, followed by the asterisk character (**/***), and end with the asterisk character, followed with the slash character (***/**). Because the comment is completely delimited with these characters, a comment can span multiple lines.

Note that C style comments cannot be nested. For example, the following is not valid because the entire comment ends with the first ***/**:

```
/* This is the start of a comment.  
   This is still part of the comment.  
/* This is an incorrect attempt to nest a comment. */  
   This is no longer in any comment. */
```

Figure A.21 Syntax for C style comments

C++ style comments

C++ style comments start with two slash characters (//) and are no longer than one line in length. To have one logical comment span multiple lines, each line must start with the // pair.

For example:

```
// This is the start of a comment. The next line
// is a new comment line, even though it is
// logically part of the previous comment.
```

Figure A.22 Syntax for C++ style comments

Shell style comments

Shell style (also known as Perl style) comments start with the # character and are no longer than one line in length.

For example:

```
# This is the start of a comment. The next line
# is a new comment line, even though it is logically
# part of the previous comment.
```

Figure A.23 Syntax for shell style comments

Sample 3DNS Controller configuration file

The following is an example of a 3DNS Controller configuration file. Note that very few global parameters are listed. This is because you do not need to include each global parameter; you need only include those parameters for which you want to specify a value other than the default.

The following sample file contains examples of common configurations and each load balancing mode. (This sample file is several pages long.)

```
#
# Sample /etc/wideip.conf
#
# Related files are:
# /etc/named.conf
# /var/namedb/db.wip.domain.com
#

globals {
    prober          12.168.101.2    // Default prober is New York 3DNS
    encryption      yes             // Encrypt iQuery
    paths_noclobber  yes             // Don't overwrite metrics with
                                    // zeroed results
    path_ttl        2400            // Extend the life of path metrics
    rtt_probe_dynamic yes           // Switch to tcp probing if icmp
                                    // fails
    multiplex_iq     yes            // Source port is same as dest. port
                                    // for iQuery
    use_alternate_iq_port yes        // Use IANA registered port for
                                    // iQuery
}

include root_in      "/var/3dns/etc"
include root_out     "/var/3dns/etc"

/** Define 3 Datacenters: New York, Los Angeles, and Tokyo */

datacenter {
    name      "New York"
    3dns      192.168.101.2
    bigip     192.168.101.40
    host      192.168.105.40
}

datacenter {
    name      "Los Angeles"
    3dns      192.168.102.2
    bigip     192.168.102.40
}

datacenter {
    name      "Tokyo"
    bigip     192.168.103.40
}
```

```
    host      192.168.104.40
}

/**/ Sync Group Definition(s) /**/

sync_group {
    name      "sync"
    3dns      192.168.101.2    // New York
    3dns      192.168.102.2    // Los Angeles
}

/**/ 3DNS Definitions /**/

// New York
server {
    type      3dns
    address   192.168.101.2
    name      "3dns-newyork"
    remote {
        secure      no
        user         "root"
    }
    probe_protocol icmp
    port         53
}

// Los Angeles
server {
    type      3dns
    address   192.168.102.2
    name      "3dns-la"
    remote {
        secure      no
        user         "root"
    }
    probe_protocol tcp
    port         53
}

/**/ BIG/ip Definitions /**/

// The New York BIG/ip is behind a firewall and the virtual servers
```



```
// need to be translated
server {
    type            bigip
    address         192.168.101.40
    name            "bigip-newyork"

    remote {
        secure      yes
        user         "administrator"
    }

    # Tell 3DNS about the 2 interfaces on a BIG/ip HA
    interface {
        address      192.168.101.41
        address      192.168.101.42
    }

    # Change the number of factories doing the work at big3d
    factories {
        prober       6
        discovery    1
        snmp          1
        hops         2
    }

    vs {
        address      192.168.101.50
        service       "http"
        translate {
            address   10.0.0.50
            port      80
        }
    }
    vs {
        address      192.168.101.50:25 // smtp
        translate {
            address   10.0.0.50:25
        }
    }
    vs {
        address      192.168.101.60:21 // ftp
        translate {
            address   10.0.0.60:21
        }
    }
}
```

```

    }
    vs {
        address      192.168.101.60
        port          80
        translate {
            address    10.0.0.60
            port        80
        }
    }
    vs {
        address      192.168.101.70
        port          443
        translate {
            address    10.0.0.70
            port        443
        }
    }
    vs {
        address      192.168.101.70:80 // http
        translate {
            address    10.0.0.70:80
        }
    }
}

// Los Angeles
server {
    type          bigip
    address        192.168.102.40
    name           "bigip-la"
    remote {
        secure      no
    }
    vs {
        address      192.168.102.50:80
    }
    vs {
        address      192.168.102.50:25
    }
    vs {
        address      192.168.102.60:21
    }
    vs {
        address      192.168.102.60:443
    }
}

```

```
}
vs {
    address      192.168.102.60:80
}
vs {
    address      192.168.102.70:80
}
}

// Tokyo
server {
    type          bigip
    address       192.168.103.40
    name          "bigip-tokyo"
    vs {
        address   192.168.103.50:80
    }
    vs {
        address   192.168.103.50:25
    }
    vs {
        address   192.168.103.60:21
    }
    vs {
        address   192.168.103.60:80
    }
    vs {
        address   192.168.103.70:80
    }
}

/** Host Definitions */

server {
    type          host
    address       192.168.104.40
    name          "host-tokyo"
    probe_protocol icmp
    port          53
    snmp {
        agent      ucd
        community   "public"
        version     1
    }
}
```

```
vs {
    address      192.168.104.50:25
}
vs {
    address      192.168.104.50:80
}
vs {
    address      192.168.104.50:443
}
}

server {
    type          host
    address       192.168.105.40
    name          "host-la"
    probe_protocol tcp
    prober        192.168.103.40    // Use the prober in Tokyo
    port          80
    snmp {
        agent      solstice
        community   "3dns"
        version     2
    }
    vs {
        address      192.168.105.50
        port          80
        probe_protocol tcp
    }
    vs {
        address      192.168.105.50:25
        probe_protocol tcp
    }
    vs {
        address      192.168.105.60:443
        probe_protocol icmp
    }
    vs {
        address      192.168.105.60:80
        probe_protocol icmp
    }
}

/*** WIP Definitions ***/
```

```
// Two pools with a production rule
wideip {
    address          192.168.101.50
    service          "http"
    name             "www.wip.domain.com"
    ttl              60          // increase the domain default ttl
    qos_coeff {
        rtt          21
        hops         0
        completion_rate 7
        packet_rate   5
        topology      1
    }
    pool {
        name          "Pool_1"
        ratio          2          // applies to pool_lbmode == ratio

        preferred      leastconn
        alternate       ratio

        // Pool rules can start anywhere after the pool name
        // and before the virtual servers.

        // Add some special rules to switch lbmodes:
        //   Weekday 6am-5pm: qos
        //   Weekday 5pm-6am: leastconn
        //   Weekend       : packet_rate

        /*** If a weekday ***/
        rule "myRule1"
        if(day != "sat" && day != "sun") {

            /*** If during business hours 6am-5pm, do QOS ***/
            rule "myRule2"
            if(preferred != "qos" && (time >= "6:00" && time <=
"17:00")) {

                // switch the lbmode and log a message that it happened
                preferred qos
                log("Switching preferred to $preferred")
            }

            /*** Otherwise, do least connections ***/
            else {
```

```
        rule "myRule3"
        if(preferred != "leastconn") {
            preferred leastconn
            log("Switching preferred to $preferred")
        }
    }
}

/** If weekend, switch to packet rate */
else {
    rule "weekendPolicy"
    if(preferred != "packet_rate") {
        preferred packet_rate
        log("Switching preferred to $preferred")
    }
}

address      192.168.101.50    ratio 2
address      192.168.102.50    ratio 1
address      192.168.103.50    ratio 1

}
pool {
    name          "Pool_2"
    ratio          1
    preferred      rr
    address        192.168.102.60    ratio 2
    address        192.168.103.60    ratio 1
}
}

// Global availability
wideip {
    address        192.168.101.60
    port           80 // http
    name           "cgi.wip.domain.com"
    pool {
        name          "mypool"
        preferred      ga
        address        192.168.101.60
        address        192.168.102.60
        address        192.168.103.60
    }
}
}
```

```
// Round robin pool load balancing between bigip and hosts
// This site runs a catalog and shopping cart and only wishes
// to send client to a datacenter if services are up on both
// ports 80 and 443.
wideip {
    address          192.168.101.70
    port             80 // http
    port_list        80 443          // e-commerce
    name             "ssl.wip.domain.com"
    pool_lbmode      rr
    pool {
        name         "bigip_pool"
        ratio        2
        preferred     qos
        alternate     ratio
        address       192.168.101.70    ratio 7
        address       192.168.102.60    ratio 2
    }
    pool {
        name         "host_pool"
        ratio        1
        preferred     ratio
        address       192.168.104.50    ratio 2
        address       192.168.105.60    ratio 1
    }
}

// Mixing hosts and BIG/ip virtual servers
// Ratio pool load balancing between bigip and hosts
wideip {
    address          192.168.102.50
    service          "smtp"
    name             "mx.wip.domain.com"
    alias            "mail.wip.domain.com"
    pool_lbmode      ratio
    pool {
        name         "pool_1"
        ratio        3
        preferred     rtt
        alternate     random
        address       192.168.101.50
        address       192.168.102.50
```

```

        address      192.168.103.50
    }
    pool {
        name          "pool_2"
        ratio          1
        preferred      ratio
        address        192.168.104.50    ratio 2
        address        192.168.105.50    ratio 1
    }
}

// Least connections with ratio as an alternate
wideip {
    address          192.168.102.60
    service          "ftp"
    name             "ftp.wip.domain.com"
    pool {
        name          "main_pool"
        preferred      leastconn
        alternate      ratio
        address        192.168.101.60    ratio 2
        address        192.168.102.60    ratio 4
        address        192.168.103.60
    }
}

// Preferred set to QOS with attributes dynamic ratio and ldns_rr
// Alternate set to leastconn
wideip {
    address          192.168.103.70:80
    name             "www2.wip.domain.com"
    pool {
        name          "pool_1"
        dynamic_ratio  yes
        rr_ldns        yes
        rr_ldns_limit  10 // Return up to 10 available virtual servers
        preferred      qos
        alternate      leastconn
        address        192.168.101.60
        address        192.168.102.60
        address        192.168.103.70
    }
}

```



```

// Global availability pool load balancing between bigip
datacenters
// with specialized use of preferred, alternate, and fallback load
// balancing methods null and return_to_dns.
wideip {
    address          192.168.102.70
    port              80
    name              "www.domain.com"
    alias             "home.domain.com"
    ttl               120
    pool_lbmode       ga
    pool {
        name          "New York"
        ratio          2
        preferred      leastconn
        alternate       null
        fallback        null          // null here allows fail over to
next pool
        address        192.168.101.50
        address        192.168.101.60
        address        192.168.101.70
    }
    pool {
        name          "Los Angeles"
        ratio          1
        preferred      leastconn
        alternate       null
        fallback        null // null here allows fail over to next pool
        address        192.168.102.50
        address        192.168.102.60
        address        192.168.102.70
    }
    pool {
        name          "Tokyo"
        ratio          1
        preferred      leastconn
        alternate       null
#    fallback        return_to_dns // (this is the fallback default)
        address        192.168.103.50
        address        192.168.103.60
        address        192.168.103.70
    }
}
}

```

```
// Round trip time load balancing with topology as alternate load
// balancing (see topology below)
wideip {
    address      192.168.103.60
    port         80
    name         "ntp.wip.domain.com"
    pool {
        name      "poolA"
        preferred rtt
        alternate topology
        address   192.168.101.60 // New York
        address   192.168.102.60 // Los Angeles
        address   192.168.103.60 // Tokyo
    }
}

/** Topological distribution and access control */

topology {
    acl_threshold 5
    probe_threshold 5
    limit_probes yes
    longest_match yes

// server/mask    ldns/mask    score

////////////////////////////////////
// North American LDNS's:
//   198.0.0.0/8
//   199.0.0.0/8

// North America Priority List
//
// 1. New York
// 2. L.A.
// 3. Tokyo

// New York
192.168.101.0/24    198.0.0.0/8    30
192.168.101.0/24    199.0.0.0/8    30

// Los Angeles
```

```

192.168.102.0/24    198.0.0.0/8    20
192.168.102.0/24    199.0.0.0/8    20

// Tokyo
192.168.103.0/24    198.0.0.0/8    10
192.168.103.0/24    199.0.0.0/8    10

////////////////////////////////////
// South American LDNS's:
//   200.0.0.0/8
//   201.0.0.0/8

// South America Priority List
//
// 1. Tokyo
// 2. L.A.
// (New York excluded by acl_threshold)

// Tokyo
192.168.103.0/24    200.0.0.0/8    30
192.168.103.0/24    201.0.0.0/8    30

// Los Angeles
192.168.102.0/24    200.0.0.0/8    20
192.168.102.0/24    201.0.0.0/8    20

// New York
192.168.101.0/24    200.0.0.0/8    0
192.168.101.0/24    201.0.0.0/8    0

////////////////////////////////////
// Wildcard List Record
//
// By default, if a list record is not found in the
// topology map for an LDNS, the score is assumed to
// be 0. By including the following "wildcard" list
// record, all other LDNS's (not North or South America
// as specified above) are assigned a score of 1 so
// the acl_threshold does not indicate that the
// virtual servers are down.

0.0.0.0/0    0.0.0.0/0    1

```

```
}

/*
 * Some global production rules
 */

rule "everyExample"
every(60) {
    log("We should do something nifty every minute")
}

rule "whenExample"
when(SIGHUP) {
    log("System was just restarted (i.e. ndc reload)")
}
```

Sample 3DNS Controller configuration file using include files

The following examples show the same sample configuration as shown in the previous section, but with the incorporation of include files.

Sample wideip.conf using include files

```
#
# Sample /etc/wideip.conf using include files
#
include root_in           "/var/3dns/etc"
include root_out          "/var/3dns/etc"
include global            "globals.conf"
include datacenter        "datacenters.conf"
include sync_group        "sync_groups.conf"
include server            "3dns.conf"
include server            "bigips.conf"
include server            "hosts.conf"
include wideip            "wideips.conf"
include 3dscrip           "prodrules.conf"
```

```
include topology "topology.conf"
```

Sample globals.conf

```
// globals.conf
globals {
    prober      192.168.101.2    // Default prober is New York 3DNS
    encryption  yes              // Encrypt iQuery
    paths_noclobber yes         // Don't overwrite metrics with
                                // zeroed results
    path_ttl    400              // Extend the life of path metrics
    rtt_probe_dynamic yes       // Switch to tcp probing if icmp
                                // fails
    multiplex_iq yes            // Source port is same as dest.
                                // port for iQuery
    use_alternate_iq_port yes   // Use IANA registered port for
                                // iQuery
}
```

Sample datacenters.conf

```
// datacenters.conf
/** Define 3 Datacenters: New York, Los Angeles, and Tokyo */
datacenter {
    name      "New York"
    3dns      192.168.101.2
    bigip     192.168.101.40
    host      192.168.105.40
}
datacenter {
    name      "Los Angeles"
    3dns      192.168.102.2
    bigip     192.168.102.40
}
datacenter {
    name      "Tokyo"
    bigip     192.168.103.40
    host      192.168.104.40
}
```

Sample sync_groups.conf

```
// sync_groups.conf
sync_group {
    name      "sync"
    3dns      192.168.101.2    // New York
}
```

```
    3dns    192.168.102.2    // Los Angeles
}
```

Sample 3dns.conf

```
// 3dns.conf
// New York
server {
    type            3dns
    address         192.168.101.2
    name            "3dns-newyork"
    remote {
        secure      no
        user         "root"
    }
    probe_protocol  icmp
    port           53
}
// Los Angeles
server {
    type            3dns
    address         192.168.102.2
    name            "3dns-la"
    remote {
        secure      no
        user         "root"
    }
    probe_protocol  tcp
    port           53
}
```

Sample bigips.conf

```
bigips.conf
// The New York BIG/ip is behind a firewall and the virtual servers
// need to be translated
server {
    type            bigip
    address         192.168.101.40
    name            "bigip-newyork"
    remote {
        secure      yes
        user         "administrator"
    }
    # Tell 3DNS about the 2 interfaces on a BIG/ip HA
    interface {
```

```

        address      192.168.101.41
        address      192.168.101.42
    }
    # Change the number of factories doing the work at big3d
    factories {
        prober        6
        discovery     1
        snmp           1
        hops           2
    }
    include vs "bigip_ny/vs.conf"
    include path "bigip_ny/path.conf"
}
// Los Angeles
server {
    type            bigip
    address          192.168.102.40
    name             "bigip-la"
    remote {
        secure       no
    }
    include vs "bigip_la/vs.conf"
    include path "bigip_la/path.conf"
}
// Tokyo
server {
    type            bigip
    address          192.168.103.40
    name             "bigip-tokyo"
    vs {
        address      192.168.103.50:80
    }
    include vs "bigip_tokyo/vs.conf"
    include path "bigip_tokyo/path.conf"
}

```

Sample bigip_ny/vs.conf

```

// bigip_ny/vs.conf
vs {
    address      192.168.101.50
    service      "http"
    translate {
        address   10.0.0.50
    }
}

```

```

        port      80
    }
}
vs {
    address      192.168.101.50:25 // smtp
    translate {
        address   10.0.0.50:25
    }
}
vs {
    address      192.168.101.60:21 // ftp
    translate {
        address   10.0.0.60:21
    }
}
vs {
    address      192.168.101.60
    port         80
    translate {
        address   10.0.0.60
        port      80
    }
}
vs {
    address      192.168.101.70
    port         443
    translate {
        address   10.0.0.70
        port      443
    }
}
vs {
    address      192.168.101.70:80 // http
    translate {
        address   10.0.0.70:80
    }
}
}

```

Sample bigip_la/vs.conf

```

// bigip_la/vs.conf
vs {
    address      192.168.102.50:80
}
vs {

```



```

    address      192.168.102.50:25
  }
  vs {
    address      192.168.102.60:21
  }
  vs {
    address      192.168.102.60:443
  }
  vs {
    address      192.168.102.60:80
  }
  vs {
    address      192.168.102.70:80
  }

```

Sample bigip_tokyo/vs.conf

```

// bigip_tokyo/vs.conf
vs {
  address      192.168.103.50:25
}
vs {
  address      192.168.103.60:21
}
vs {
  address      192.168.103.60:80
}
vs {
  address      192.168.103.70:80
}

```

Sample hosts.conf

```

// hosts.conf
server {
  type          host
  address       192.168.104.40
  name          "host-tokyo"
  probe_protocol icmp
  port          53
  snmp {
    agent        ucd
    community    "public"
    version      1
  }
}
include vs "host_tokyo/vs.conf"

```

```
include path "host_tokyo/path.conf"
}
server {
    type            host
    address         192.168.105.40
    name            "host-la"
    probe_protocol  tcp
    prober          192.168.103.40    // Use the prober in Tokyo
    port            80
    snmp {
        agent       solstice
        community    "3dns"
        version      2
    }
}
include vs "host_la/vs.conf"
include path "host_la/path.conf"
}
```

Sample host_tokyo/vs.conf

```
// host_tokyo/vs.conf
vs {
    address      192.168.104.50:25
}
vs {
    address      192.168.104.50:80
}
vs {
    address      192.168.104.50:443
}
```

Sample host_la/vs.conf

```
// host_la/vs.conf
vs {
    address      192.168.105.50
    port         80
    probe_protocol tcp
}
vs {
    address      192.168.105.50:25
    probe_protocol tcp
}
vs {
    address      192.168.105.60:443
    probe_protocol icmp
}
```

```

}
vs {
    address      192.168.105.60:80
    probe_protocol icmp
}

```

Sample wideips.conf

```

// wideips.conf
// Two pools with a production rule
wideip {
    address      192.168.101.50
    service      "http"
    name         "www.wip.domain.com"
    ttl          60      // increase the domain default ttl
    qos_coeff {
        rtt      21
        hops     0
        completion_rate 7
        packet_rate 5
        topology 1
    }
    pool {
        name      "Pool_1"
        ratio     2      // applies to pool_lbmode == ratio
        preferred leastconn
        alternate ratio
        address   192.168.101.50    ratio 2
        address   192.168.102.50    ratio 1
        address   192.168.103.50    ratio 1
    }
    pool {
        name      "Pool_2"
        ratio     1
        preferred rr
        address   192.168.102.60    ratio 2
        address   192.168.103.60    ratio 1
    }
}

// Global availability
wideip {
    address      192.168.101.60
    port         80 // http
    name         "cgi.wip.domain.com"
}

```

```

    pool {
        name          "mypool"
        preferred      ga
        address        192.168.101.60
        address        192.168.102.60
        address        192.168.103.60
    }
}

// Round robin pool load balancing between bigip and hosts
// This site runs a catalog and shopping cart and only wishes
// to send client to a datacenter if services are up on both
// ports 80 and 443.
wideip {
    address          192.168.101.70
    port             80 // http
    port_list        80 443          // e-commerce
    name             "ssl.wip.domain.com"
    pool_lbmode      rr
    pool {
        name          "bigip_pool"
        ratio          2
        preferred      qos
        alternate      ratio
        address        192.168.101.70    ratio 7
        address        192.168.102.60    ratio 2
    }
    pool {
        name          "host_pool"
        ratio          1
        preferred      ratio
        address        192.168.104.50    ratio 2
        address        192.168.105.60    ratio 1
    }
}

// Mixing hosts and BIG/ip virtual servers
// Ratio pool load balancing between bigip and hosts
wideip {
    address          192.168.102.50
    service          "smtp"
    name             "mx.wip.domain.com"
    alias            "mail.wip.domain.com"
    pool_lbmode      ratio

```

```

    pool {
        name            "pool_1"
        ratio            3
        preferred        rtt
        alternate        random
        address          192.168.101.50
        address          192.168.102.50
        address          192.168.103.50
    }
    pool {
        name            "pool_2"
        ratio            1
        preferred        ratio
        address          192.168.104.50    ratio 2
        address          192.168.105.50    ratio 1
    }
}

// Least connections with ratio as an alternate
wideip {
    address            192.168.102.60
    service            "ftp"
    name               "ftp.wip.domain.com"
    pool {
        name            "main_pool"
        preferred        leastconn
        alternate        ratio
        address          192.168.101.60    ratio 2
        address          192.168.102.60    ratio 4
        address          192.168.103.60
    }
}

// Preferred set to QOS with attributes dynamic ratio and ldns_rr
// Alternate set to leastconn
wideip {
    address            192.168.103.70:80
    name               "www2.wip.domain.com"
    pool {
        name            "pool_1"
        dynamic_ratio    yes
        rr_ldns          yes
        rr_ldns_limit    10 // Return up to 10 available virtual servers
        preferred        qos
    }
}

```

```

        alternate    leastconn
        address      192.168.101.60
        address      192.168.102.60
        address      192.168.103.70
    }
}

// Global availability pool load balancing between bigip
// datacenters with specialized use of preferred, alternate, and
// fallback load balancing methods null and return_to_dns.
wideip {
    address          192.168.102.70
    port             80
    name             "www.domain.com"
    alias            "home.domain.com"
    ttl              120
    pool_lbmode      ga
    pool {
        name         "New York"
        ratio        2
        preferred     leastconn
        alternate     null
        fallback      null // null here allows fail over to next pool
        address       192.168.101.50
        address       192.168.101.60
        address       192.168.101.70
    }
    pool {
        name         "Los Angeles"
        ratio        1
        preferred     leastconn
        alternate     null
        fallback      null // null here allows fail over to next pool
        address       192.168.102.50
        address       192.168.102.60
        address       192.168.102.70
    }
    pool {
        name         "Tokyo"
        ratio        1
        preferred     leastconn
        alternate     null
#        fallback     return_to_dns // (this is the fallback default)
        address       192.168.103.50
    }
}

```

```

        address      192.168.103.60
        address      192.168.103.70
    }
}

// Round trip time load balancing with topology as alternate load
// balancing (see topology below)
wideip {
    address      192.168.103.60
    port         80
    name         "ntp.wip.domain.com"
    pool {
        name      "poolA"
        preferred rtt
        alternate topology
        address   192.168.101.60    // New York
        address   192.168.102.60    // Los Angeles
        address   192.168.103.60    // Tokyo
    }
}

```

Sample prodrules.conf

```

// prodrules.conf
// Pool rules can start anywhere after the pool name
// and before the virtual servers.

// Add some special rules to switch lbmodes:
//   Weekday 6am-5pm: qos
//   Weekday 5pm-6am: leastconn
//   Weekend      : packet_rate

/**** If a weekday ****/
rule "myRule1"
if(day != "sat" && day != "sun") {

    /**** If during business hours 6am-5pm, do QOS ****/
    rule "myRule2"
    if(preferred != "qos" && (time >= "6:00" && time <=
"17:00")) {
        // switch the lbmode and log a message that it happened
        preferred qos
        log("Switching preferred to $preferred")
    }
}

```

```

    /*** Otherwise, do least connections ***/
    else {
        rule "myRule3"
        if(preferred != "leastconn") {
            preferred leastconn
            log("Switching preferred to $preferred")
        }
    }
}

/*** If weekend, switch to packet rate ***/
else {
    rule "weekendPolicy"
    if(preferred != "packet_rate") {
        preferred packet_rate
        log("Switching preferred to $preferred")
    }
}
}

```

Sample topology.conf

```

// topology.conf
topology {
    acl_threshold 5
    probe_threshold 5
    limit_probes yes
    longest_match yes

// server/mask    ldns/mask    score

////////////////////////////////////
// North American LDNS's:
// 198.0.0.0/8
// 199.0.0.0/8

// North America Priority List
//
// 1. New York
// 2. L.A.
// 3. Tokyo

    // New York

```



```
192.168.101.0/24    198.0.0.0/8    30
192.168.101.0/24    199.0.0.0/8    30

// Los Angeles
192.168.102.0/24    198.0.0.0/8    20
192.168.102.0/24    199.0.0.0/8    20

// Tokyo
192.168.103.0/24    198.0.0.0/8    10
192.168.103.0/24    199.0.0.0/8    10

////////////////////////////////////
// South American LDNS's:
//   200.0.0.0/8
//   201.0.0.0/8

// South America Priority List
//
// 1. Tokyo
// 2. L.A.
// (New York excluded by acl_threshold)

// Tokyo
192.168.103.0/24    200.0.0.0/8    30
192.168.103.0/24    201.0.0.0/8    30

// Los Angeles
192.168.102.0/24    200.0.0.0/8    20
192.168.102.0/24    201.0.0.0/8    20

// New York
192.168.101.0/24    200.0.0.0/8    0
192.168.101.0/24    201.0.0.0/8    0

////////////////////////////////////
// Wildcard List Record
//
// By default, if a list record is not found in the
// topology map for an LDNS, the score is assumed to
// be 0. By including the following "wildcard" list
// record, all other LDNS's (not North or South America
// as specified above) are assigned a score of 1 so
// the acl_threshold does not indicate that the
```

```
// virtual servers are down.

    0.0.0.0/0    0.0.0.0/0    1

}

/*
 * Some global production rules
 */

rule "everyExample"
every(60) {
    log("We should do something nifty every minute")
}

rule "whenExample"
when(SIGHUP) {
    log("System was just restarted (i.e. ndc reload)")
}
```

Understanding cur_ values

You may notice several **cur_** values in your **wideip.conf** file. The purpose of **cur_** values is to pre-load the database with previously collected statistics and metrics. The collected statistics and metrics are useful if you want to quickly restart a 3DNS Controller without a temporary loss of intelligence.

Do not edit these statements unless you are a very experienced 3DNS Controller user, or you are instructed to do so by F5 technical support.

How `cur_` values are used

You may notice **`cur_`** values in **`server`**, **`vs`**, **`path`**, or **`wideip`** definitions. Examples for each type of definition follow.

Example: server definition

```
// New York BIG/ip Controller
server {
    type bigip
    address 192.168.101.40
    cur_packet_rate 139
    cur_ok 1
    [virtual server definitions]
}
```

Figure A.24 *`cur_` values in a server definition*

In the above example, the **`cur_`** values indicate the following.

Parameter	Description
<code>cur_packet_rate</code>	Indicates the number of packets per second sent during the last sample period.
<code>cur_ok</code>	Indicates the state of the specified server. The options are: 1 (Up), 2 (Down), 3 (Waiting), 4 (Alert), and 5 (Panic).

Table A.27 *`cur_` values in a server definition*

Example: vs definition

```
vs {  
  address 192.168.102.50:80  
  cur_serv_cnt 1  
  cur_connections 0  
  cur_picks 39  
  cur_refreshes 783  
}
```

Figure A.25 cur_ values in a virtual server definition

In the above example, the **cur_** values indicate the following:

Parameter	Description
cur_nodes_up	Indicates the number of active servers serving the specified virtual server.
cur_connections	Indicates the number of connections to the specified virtual server.
cur_picks	Indicates the number of times the specified virtual server was returned by the 3DNS Controller.
cur_refreshes	Indicates the number of times the server and connection counts were refreshed with new data.

Table A.28 cur_ values in a virtual server definition

Example: path definition

```
path {  
  address 10.25.50.100 // LDNS  
  cur_rtt 102382  
  cur_completion_rate 10000  
  cur_picks 239  
  cur_accesses 302  
}
```

Figure A.26 *cur_ values in a path definition*

In the above example, the **cur_** values indicate the following:

Parameter	Description
cur_rtt	Indicates the round trip time (RTT) , which is a calculation of the time (in microseconds) that the specified machine takes to respond to a probe issued by the 3DNS Controller.
cur_completion_rate	Indicates the percentage of completed packets versus lost packets, using this equation: $[1 - (\text{packets received} / \text{sent})] \times 10000$.
cur_picks	Indicates the number of times this path's data resulted in the virtual server being chosen for a connection. This only applies if a wide IP is doing dynamic load balancing (using path data).
cur_accesses	Indicates the number of times this path was considered when performing dynamic load balancing.

Table A.29 *cur_ values in a path definition*

Example: wide IP definition

```
wideip {  
  address 192.168.102.70  
  name "www.domain.com"  
  port 80  
  cur_preferred 143982  
  cur_alternate 108090  
  cur_fallback 130094  
  cur_returned_to_dns 23872  
  [virtual server definitions]  
}
```

Figure A.27 *cur_ values in a wide IP definition*

In the above example, the **cur_** values indicate the following:

Parameter	Description
cur_preferred	Indicates the number of times the specified wide IP was resolved by the preferred load balancing mode.
cur_alternate	Indicates the number of times the specified wide IP was resolved by the alternate load balancing mode.
cur_fallback	Indicates the number of times the specified wide IP was resolved by the fallback load balancing mode.
cur_returned_to_dns	Indicates the number of times the specified wide IP did not find a suitable virtual server to return using the preferred , alternate , or fallback load balancing modes. In this situation, the 3DNS Controller returns the wide IP key (fallback address) as specified in the zone file.

Table A.30 *cur_ values in a wide IP definition*

◆ **Tip**

*To find out how many times the 3DNS Controller received resolution requests for this wide IP, add the values for **cur_preferred**, **cur_alternate**, and **cur_fallback**.*



B

3DNS Controller Utilities and Scripts



Utilities

The 3DNS Controller includes several utilities and scripts. These utilities and scripts allow you to configure the DNS, and the various features of the 3DNS Controller.

3dparse

The **3dparse** utility parses and verifies the syntax of the 3DNS configuration file (**wideip.conf**). You can use it to verify syntax after making any changes to **wideip.conf**, before running **named**.

The **3dparse** utility can be used to validate configuration syntax. **3dparse** checks global value ranges and to ensure each virtual server is configured on a BIG/ip Controller or other host machine. The **3dparse** utility also checks dependencies. For example, TTL values (like **bigip_ttl**) must be greater than their corresponding timer values (like **timer_get_bigip_data**).

Use the following syntax with **3dparse**:

```
3dparse [-help] [-v] [-o] [-if <string>] [-of <string>] \  
        [-sf <string>] [-trim] [-flat] [-actions] [-d] [-s] [-vl] \  
        [-picky]
```

The options for **3dparse** are:

-help

Displays the list of available options.

-v

Displays the version information.

-o

Writes the configuration information to the output file.

-if <string>

Specifies a file name for the input file. If you do not use this option, 3dparse uses the default input file, **/etc/wideip.conf**.

-of <string>

Specifies a file name for the output file and writes the configuration information to that file. The default file name is **./3dparse.out**.

-sf <string>

Specifies the path for output status file. The default is **stdout**.

-trim

Writes all path and local DNS information to include files. By default, path and local DNS information is sent to include files.

-flat

Writes all information to one output file, regardless of whether include files are used.

-actions

Writes all actions (production rules) to the output file. By default, production rules are not written to the output file.

-d

Simulates a SIGINT, which has the effect of writing status files to **/var/run**.

-s

Simulates receiver behavior when loading.

-vl

Turns on **syslog** verbosity and path loading.

-picky

Specifies that any validation errors are not automatically corrected.

Example of a 3dparse command

The following example shows a **3dparse** command. The bold typeface indicates the typed command.

```
bighost:~# 3dparse -o
3dparse: Initializing ...
3dparse: Parsing /etc/wideip.conf
3dparse: Dumping ./3dparse.out
3dparse: SUCCESS
```

sod-named

Use the **sod-named** utility to ensure that a version of **named** is always running on the 3DNS Controller. Always run **sod-named** after rebooting the 3DNS Controller.

If **sod-named** is running, do not manually start **named**, as this causes the currently running **named** process to stop.

sod-named performs the following functions:

- ❖ Starts and watches new **named** and **syncd** processes if **named** and **syncd** are not running when **sod-named** is started.
- ❖ Monitors any running **named** and **syncd** processes.
- ❖ Starts new **named** and **syncd** processes if the watched **named** and **syncd** processes stop.
- ❖ Keeps secure any dumped **named** core files by renaming the core file and adding a timestamp suffix. **sod-named** then compresses the core file.
- ❖ Presents an error message if you attempt to start more than one **sod-named** process.
- ❖ Logs an emergency message if the **named** process runs for less than one hour before stopping, ten times in a row; this behavior usually indicates a serious problem with **named**. You can use the **-restart** or **-seconds** arguments when you start **sod-named** to change the time parameters. These arguments are described later in this section.
- ❖ Parses **named.conf** to find the **directory** command in order to find in which directory to run and where to dump and find **named** cores. If more than one directory command is found in **named.conf**, **sod-named** uses the last one it finds.

If you are using a **ps** command followed by a **grep named** command to find all **named** process on a 3DNS Controller, add the **-ww** argument to the **ps** command. This causes **ps** to print out long lines, ensuring that **sod-named** appears in the output.

A 3DNS Controller does not have to use **sod-named**. You can instead use **named** and **ndc**. See *named*, on page B-5, and *Configuring syslog for 3DNS messages*, on page B-7.

Use the following syntax with **sod-named**:

```
sod  [ -tty00 ] [ -tty01 ] [ -ethernet ]  
    [ -force_master ] [ -force_slave ] [ -force_active ]  
    [ -force_standby ] [ -restart # of restarts ]  
    [ -seconds # of secs ] [ -config named.conf path ]  
    [ -no-watch-named ] [ -no-watch-syncd ]  
    [ -shared IP address ] [ -peer IP address ]
```

3ndc

3ndc allows the name server administrator to send various signals to the name server, or to restart it.

◆ Note

*The name daemon control interface command **ndc** is now an alias for **3ndc**.*

The syntax for **3ndc** is as follows:

3ndc directive [...]

When you use **3ndc**, you can specify directives. Directives are not required. Directives for **3ndc** include:

status

Display the current status of **named**, **sod_named**, and **syncd** as shown by ps(1).

dumpdb

Write **named**'s database and cache to **/var/tmp/named_dump.db**. It uses the INT signal.

reload

Checks the serial numbers of all primary and secondary zones and reloads those that have changed. Uses the HUP signal.

stats

Writes statistics to **/var/tmp/named.stats**. Uses the IOT or ABRT signal.

trace

Increments the tracing level by one. Whenever the tracing level is not zero, trace information is written to **/var/tmp/named.run**. Higher tracing levels result in more detailed information. Uses the USR1 signal.

notrace | cmd

Rereads the **/var/run/widip.cmd** file and sets its tracing level to zero. The **/var/tmp/named.run** closes if it is open. Uses the USR2 signal. Using **notrace** or **cmd** has the same effect, and can be used in addition to using the same argument with **ndc**.

querylog

Toggles the query logging feature which, while on, results in a syslog(3) entry for each incoming query. It uses the WINCH signal. Note that query logging consumes log file space. This directive may also be given as **qrylog**.

start

Starts **sod-named**, if it is not running. **sod-named** starts **named** and **syncd**. If **named** and **syncd** processes are already running, **sod-named** starts and watches the current **named** and **syncd** processes.

stop

Stops **sod-named**, **named**, and **syncd**, if they are running.

restart

Stops and restarts **sod-named**, **named**, and **syncd**.

named

named is the Internet domain name server. If no arguments are specified, **named** opens the default boot file (**/etc/named.conf**), reads any initial data, and listens for queries.

named uses the following syntax:

```
named [ -(b|c) <config_file> ] [ -d <debuglevel> ] [ -f ] \  
[ -g <group_name> ] [ -p <port#> ] [ -q ] [ -r ] \  
[ -t <directory> ] [ -u <user_name> ] [ -v ] [ -w <directory> ] \  
[ config_file ]
```

The options for **named** include:

-b

Specifies an alternate boot file. This argument is overridden by any configuration file which is specified at the end of the command line. The default value is **/etc/named.conf**.

-d

Prints debugging information. The number specified after this option determines the level of printed messages.

-f

Runs the process in the foreground.

-g

Specifies which group the server should run as after it initializes. You can specify a group name or a numeric group ID.

-p

Use the specified remote port number; this is the port number to which **named** sends queries. The default value is the standard port number as returned by the **getservby-name** command for the service domain. In earlier versions of **named**, the syntax **-p port#[/localport#]** was supported. The first port was used when contacting remote servers, and the second one was the service port bound by the local instance of **named**. The current usage is equivalent to the old usage without the **localport#** specified; this functionality can be specified with the listen-on clause of the configuration file's **options** statement.

-q

Traces all incoming queries if **named** was compiled with the **QRYLOG defined** command. Note that this option is deprecated in favor of the boot file directive: **options query-log**.

-r

Turns off recursion on the server. Answers can come only from local (primary or secondary) zones. This option can be used on root servers. Note that this option is deprecated in favor of the boot file directive: **options no-recursion**.

-t

Specifies the directory the server should **chroot(2)** into as soon as it finishes processing command line arguments.

-u

Specifies the user the server should run as after it initializes. You can specify a user name or a numeric user ID. If you did not use the **-g** option, the group ID used is the primary group of the specified user—**initgroups(3)**—is called, so all of the user's groups are available to the server.

-v

Displays the version information.

-w

Sets the working directory of the server. The directory clause of the configuration file's **options** statement overrides any value specified on the command line. The default working directory is the current directory.

[config_file]

Any additional argument is taken as the name of the configuration file, for compatibility with older implementations; as noted above, this argument overrides any configuration file specified by the **-b** and **-c** options. If no further argument is given, the default configuration file is used (**/etc/named.conf**).

For more information on **named**, see the **named** man page.

Configuring syslog for 3DNS messages

Although the **syslog** daemon is configured to save 3DNS Controller messages by default, the information in this section is provided in case you ever need to reconfigure your system. The lines listed in the following procedure are default entries for files shipped with a new 3DNS Controller.

Both the **big3d** agent and **named** utility use the **syslog** daemon and all messages are written to the local2 facility.

You can view the log files in the F5 Configuration utility by clicking **Log Files** in the navigation pane.

To set up 3DNS Controller logging manually

1. Add the following lines to the `/etc/syslog.conf` file.

```
local2.err           /var/log/3dns  
local6.err           /var/log/sync
```

To include warnings in normal operations, also add the following line:

```
local2.warning       /var/log/3dns
```

For full debugging, add the following line:

```
local2.debug         /var/log/3dns
```

The above lines are somewhat equivalent to:

```
local2.*             /var/log/3dns
```

As an alternative, you can use a different file to capture a session without affecting the default files. For example, you could use a line like the following:

```
local2.debug         /var/log/3dns.debug
```

To switch logging levels or specify another file name, edit the `/etc/syslog.conf` file and restart **syslogd** or issue it a **SIGHUP**.

2. Create an empty 3DNS Controller file in `/var/log` by typing the following on the command line:

```
% touch 3dns  
% touch sync
```

Note that in the above example, **3dns** is the name of the file you are creating. You can use this command to create other files for the 3DNS Controller (with different names). You need only create other 3DNS Controller files when solving configuration problems.

You must **touch** each file that you create. Continuing with the examples in step 1, type the following entry:

```
% touch 3dns.debug
```

3. Restart **syslog** by typing the following on the command line:

```
kill -HUP `cat /var/run/syslog.pid`
```

Log rotation

The 3DNS Controller's log file is called **/var/log/3dns**. The 3DNS Controller uses log rotation to keep log files from becoming overly large. A script included with the 3DNS Controller, **/etc/daily**, automatically runs each night, compressing the existing information in the log file. We do not recommend that you edit this file.

syslog.conf

The **syslog.conf** file is the configuration file for the **syslogd** program. It consists of blocks of lines separated by program specifications, with each line containing two fields:

❖ **Selector field**

Specifies the types of messages and priorities to which the line applies.

❖ **Action field**

Specifies the action to be taken if **syslogd** receives a message that matches the selection criteria.

The selector field is separated from the action field by one or more space or tab characters.

The Selector function is encoded as a facility, a period (.), and a level, with no intervening white space. Both the facility and the level are not case-sensitive.

The facility describes the part of the system generating the message, and is one of the following keywords: **auth**, **authpriv**, **cron**, **daemon**, **ftp**, **kern**, **lpr**, **mail**, **mark**, **news**, **ntp**, **syslog**, **user**, **uucp**, and **local0** through **local7**. These keywords (with the exception of **mark**) correspond to the similar LOG_ values specified to the openlog and syslog library routines.

The level describes the severity of the message. The severity levels include (from highest to lowest): **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, and **debug**. These correspond to the similar LOG_ values specified to the syslog library routine.

Each block of lines in the **syslog.conf** file is separated from the previous block by a tag. The tag is a line beginning with one of the following:

❖ **#!prog**

Used for compatibility with the previous syslogd; for example, if one is sharing **syslog.conf** files.

❖ **!prog**

Each block will be associated with calls to **syslog** from that specific program.

The action specified in the action field is taken if a message received matches the specified facility and is of the specified level (or a higher level), and if the first word in the message after the date matches the program.

To specify multiple selectors for a single action, separate each selector with a semicolon (;) character. It is important to note that each selector can modify the ones preceding it.

To specify multiple facilities for a single level, separate each selector with a comma (,) character.

An asterisk (*) can be used as a wildcard character to specify all facilities, all levels, or all programs.

The special facility **mark** receives a message at **info** priority every 20 minutes. This is not enabled by a facility field. The facility command uses the following marks:

❖ **A comma separated list of users**

Selected messages are written to those users if they are logged in.

❖ **An asterisk**

Selected messages are written to all logged-in users.

❖ **A vertical bar (|)**

The vertical bar is followed by a command to which to pipe the selected messages. The command is passed to a **/bin/sh** for evaluation, so usual shell metacharacters or input/output redirection can occur. (However, note that redirecting **stdio**

buffered output from the invoked command can cause additional delays, or even lost output data in case a logging sub-process exited with a signal.) The command itself runs with stdout and stderr redirected to `/dev/null`. Upon receipt of a SIGHUP, **syslog.conf** closes the pipe to the process. If the process did not exit voluntarily, it will be sent a SIGTERM signal after a grace period of up to 60 seconds.

The command starts only when the data that should be piped to it arrives. If the process exits later, it restarts as necessary. If you want the sub-process to get exactly one line of input only (which can be very resource-consuming if there are a lot of messages flowing quickly), you can do this by exiting after just one line of input. If necessary, a script wrapper can be written to this effect.

Unless the command is a full pipeline, you probably want to start the command with **exec** so that the invoking shell process does not wait for the command to complete.

◆ **WARNING**

*The process is started under the UID that invokes **syslogd**, usually the superuser.*

Blank lines and lines whose first non-blank character is a hash (#) character are considered to be comments, and are ignored.

Example

The following is an example of a configuration file:

```
# Log all kernel messages, authentication messages of
# level notice or higher and anything of level err or
# higher to the console.
# Don't log private authentication messages!
*.err;kern.*;auth.notice;authpriv.none /dev/console
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none /var/log/messages
```

```
# The authpriv file has restricted access.
authpriv.*                                /var/log/secure
# Log all the mail messages in one place.
mail.*                                    /var/log/maillog
# Everybody gets emergency messages, plus log them on another
# Save ftpd transactions along with mail and news
!ftpd
*. *                                      /var/log/spoolerr
```

syslogd

The **syslogd** daemon reads and logs messages to the system console, log files, other machines, and/or users as specified by its configuration file.

The **syslogd** daemon uses the following syntax:

```
syslogd [-a <allowed_peer>] [-d] [-f] [-m] [-p] [-s]
```

Options include the following:

-a <allowed_peer>

Allows **allowed_peer** to log to this syslogd using UDP datagrams. Multiple **-a** options may be specified.

Allowed_peer can be any of the following:

❖ **ipaddr/masklen[:service]**

Accepts datagrams from **ipaddr** (in the usual dotted quad notation) with **masklen** bits being taken into account when doing the address comparison. If specified, **service** is the name or number of a UDP service to which the source packet must belong. A service of ***** allows packets sent from any UDP port. The default service is "**syslog**". A missing **masklen** is substituted by the historic class A or class B netmasks if **ipaddr** belongs to the address range of class A or B, respectively, or by 24 otherwise.

❖ domainname[:service]

Accepts datagrams where the reverse address lookup yields the domainname for the sender's address. The meaning of service is described above.

❖ *domainname[:service]

Same as above, except that any source host whose name ends in domainname will get permission.

-d

Puts **syslogd** into debugging mode. This is useful for troubleshooting.

-f

Specifies the path name of an alternate configuration file; the default is **/etc/syslog.conf**.

-m

Selects the number of minutes between mark messages; the default is 20 minutes.

-p

Specifies the path name of an alternate log socket; the default is **/var/run/log**.

-s

Operates in secure mode. Does not listen for log message from remote machines.

log2mail

The **log2mail** program gathers system log messages from the **syslogd** daemon and mails a copy to each specified address. It is intended to be invoked by **syslogd** using the "|" construct in the **/etc/syslog.conf** file, as in the following example:

```
*.err,auth.notice    |/usr/sbin/log2mail root@remote.site.com
```

The **log2mail** program begins each mail message with a line of context taken from the previous mail message. The context clarifies the meaning of the "last message repeated n times" messages that are generated by **syslogd** itself.

log2mail uses this syntax:

```
log2mail [-t <interval> ]
```

One option is available:

```
-t <interval>
```

Specifies the minimum interval in seconds between consecutive mail messages. When **log2mail** receives a new log message, it checks whether **<interval>** seconds have passed since the last time it mailed a message. If at least that amount of time has passed, **log2mail** mails the new message without delay. Otherwise, it saves incoming messages and sends them later, after **<interval>** seconds have passed since the previous mail. This prevents a large number of log messages from producing many mail messages.

The default interval is 300 seconds (5 minutes).

big3d

The **big3d** utility is the 3DNS daemon which answers metrics collection questions for 3DNS.

Use the following syntax with **big3d**

```
big3d [ -foreground ] [ -help ] [ -rxbufsize value ] [ -rtt-  
timeout value ] [ -txbufsize value ] [ -version ] [ -v ] [ -keyfile file ] [ -encryption_only ] [ -max_active_probers value ] [ -max_active_scanners value ] [ -max_active_hops value ] [ -max_active_snmp value ] [ -loglevel value ]
```

The options for **big3d** are:

```
-foreground
```

Runs the process in the foreground rather than as a daemon.

```
-help
```

Prints this message.

```
-rxbufsize value
```

Represents the size of the receive socket buffer. 30720 is the default.

```
-rtt-timeout value
```


Overrides timeout value for round trip times.

-txbufsize *value*

Represents the size of the receive socket buffer. 20480 is the default.

-version

Prints the version information.

-keyfile *file*

Uses an alternative encryption key file (domestic version only).

<-encryption_only>

Accepts only encrypted messages (domestic version only).

-max_active_probers *value*

-max_active_scanners *value*

-max_active_hops *value*

-max_active_snmp *value*

Defines the maximum number of factories that can be operating when the controller is in the ACTIVE state.

-loglevel *value*

Sets the level of messages sent to syslog; notify is the default.

-item

Example

```
bighost:~# big3d
```

Files

/var/log/3dns

bigdb

The **BIG/db** database contains configuration data used by various utilities and daemons that run on a BIG/ip Controller. The database is located at **/var/f5/bigdb/user.db**. Records are stored in the

database as binary data in key/value pairs. You can access the database by using the **bigdba** command line utility. BIG/config also uses the database to access certain configuration information.

BIG/db keys

- ❖ A BIG/db key is a simple character string.
- ❖ Key names are not case sensitive.
- ❖ Key names may not contain the characters `'"', '=', '*', '{' or '}'`
- ❖ Key names may contain spaces, tabs or newline characters provided the name is always enclosed in `''`'s when it is referenced.
- ❖ Key names may begin with any valid character, and may be up to 1024 bytes in length.
- ❖ In BIG/db a key is composed of one or more subkeys delimited by `.'` characters. An example of a valid subkey is **BigIp.Test.IpAddr** where the top level key is **BigIp** with a subkey named **Test** and a subkey **IpAddr** under **Test**.

BIG/db records

BIG/db records may be of arbitrary length within the constraint of available memory; however, bigdba imposes a limit of approximately 16KB on the size of a record. Records are typically in the form of character strings, making the data easy to display and modify.

BIG/db dump files

A BIG/db dump file is a simple text file. Text files may be used to keep backup copies of the database and to load new data into the database. You can use the **bigdba** utility to dump the database to a text file and to load records into the database from a text file.

A line in the text file may be either an assignment, a comment, or an empty key specification.

An assignment has the form

```
<key>[.subkey[.subkey[...]]] =<value>
```

A comment begins with a '#' at the beginning of a line, and ends at the next line that does not begin with a '#'. A comment precedes the key that the comment describes.

An empty key is specified by a line containing only the name of the key with no assignment.

It is a good idea to keep a copy of the database in text file format in case the database is accidentally modified.

BIG/db comments

A key ending with the character sequence '#' is a comment. Any key in the database may have an associated comment record. The key name for a comment is <original key>.#' Comments may contain multiple lines of text describing the purpose of the associated key. By default comments are not displayed by the **bigdba** utility. Comments are always loaded/dumped with data to and from text files when using the **bigdba** load and dump commands.

Files

/var/f5/bigdb/user.db /var/f5/bigdb/default.txt

bigdba

The **bigdba** is a command line interface for configuring the BIG/store database. It allows you to interactively display and modify records in the BIG/store database from the command line. You can also use **bigdba** to dump the database to a text file and to load the database from a text file.

Use the following syntax for **bigdba**:

```
bigdba [-d <dump filename> ] [ -l <loadfilename> ] [ -h | -help ]  
      <database name>
```

The options for **bigdba** are:

-d filename

Dumps the entire database to the text file **filename** and exits.

-l filename

Loads the database from the text file **filename** and exits. This is an update operation. Any records in the database having the same keys as those in the file are overwritten. Any records in the database which do not have corresponding keys in the file are left unchanged.

-h | -help

Print options.

Interactive Commands

When given no options, **bigdba** opens the given database for writing, displays a >prompt, and waits for input on the command line. The database is created if it did not previously exist. Valid commands are:

print name[*] | p name[*]

Displays the value stored under the key **name**. If * is used, the values stored under all keys that begin with **name** are displayed. Key names are not case sensitive.

name = value

Stores *value* under the key **name**. If a value was previously stored under key **name**, it is overwritten.

d name[*]

Deletes the record stored under the key **name**. If ***** is used, delete the values stored under all keys that begin with **name**. If delete confirmation is on, a confirmation prompt is displayed prior to the deletion of each record.

subkey name | sk name

Adds subkey **name** to the current key level. Analogous to changing to a subdirectory using the **cd** command. The current subkey is prepended to the key argument of subsequent commands.

up | up name

Backs up one subkey, or if **name** is used, backs up through subkey **name**. Analogous to **cd**.

set confirm [on|off]

Modifies delete confirmation behavior. When delete confirmation is on, user confirmation is obtained prior to deleting each record given in a delete command. Delete confirmation is **on** by default.

set comments [on|off]

Modifies comment display. When comment display is **on**, comment records are displayed by the print command. Comment display is **off** by default.

dump file

Dumps the entire database to a text file.

load *file*

Loads the database from a text file. The database is not truncated by a load. Existing records in the database are updated by matching key values in the file.

quit | q | EOF

Exits the **bigdba** program.

help | ?

Displays help text.

Examples of a bigdba command

For example, to create, display, and delete a record called **Bigip.Tests.TestIpAddr** in the database **/var/f5/bigdb/user.db**:

```
bighost:~# bigdba /var/f5/bigdb/user.db  
Database "/var/f5/bigdb/user.db" opened.  
> Bigip.Tests.TestIpAddr = 1.2.3.4  
> p Bigip.Tests.TestIpAddr  
Bigip.Tests.TestIpAddr = "1.2.3.4"  
> d Bigip.Tests.TestIpAddr  
Confirm:  
Delete record for "Bigip.Tests.TestIpAddr" ? [Y]  
> q  
bighost:~#
```

Files

/var/f5/bigdb/user.db /var/f5/bigdb/default.txt

Notes

We recommend that you keep a backup database or text file in case the database is modified by accident.

bind2namesurfer

The **bind2namesurfer** program loads all master zone files in **/etc/named.conf** into NameSurfer.

You must have previously configured NameSurfer, and it must be running. If **named.conf file to write** is not specified, **named.conf file to parse** is used instead.

Use the following syntax with **bind2namesurfer**:

```
bind2namesurfer named.conf file to parse IP address of NameSurfer  
NameSurfer port# [named.conf file to write ]
```

Caveats

You should run this program only after you have configured NameSurfer. If you have not yet configured NameSurfer, run **config_namesurfer**. Refer to the NameSurfer documentation for errors reported when attempting to import zone files using the **nsctl** utility. This program is located in the directory **/var/f5/namesurfer/aux** and is not in the default path.

checktrap

The **checktrap.pl** utility sends traps from **syslogd** information. It reads from standard input and sends an SNMP trap when a line matches a regular expression. The specific trap sent depends on the regular expression matched. The regular expressions to trap mappings are defined in a file, which is by default **/etc/snmptrap.conf**.

The **checktrap.pl** utility is specifically designed to deal with input from **syslogd**. Normally the user puts calls to **checktrap.pl** in the **/etc/syslogd.conf** file.

The **checktrap** utility uses the following syntax:

checktrap.pl [options]

Syntax of snmptrap.conf file

Comments lines begin with **#**. All other lines are assumed to be valid for configuration lines. Valid configuration lines are of the following form:

OID REGULAR EXPRESSION DESCRIPTION

The three fields are separated by tabs.

- ❖ ***OID*** is an SNMP Object Identifier.
- ❖ ***REGULAR EXPRESSION*** is a Perl Regular Expression.
- ❖ ***DESCRIPTION*** is a string used in the description field of the trap itself.

The options for **checktrap** are:

--snmpd_conf_file= *filename*

The snmpd configuration file. This has the **trapsink --** where traps are sent to -- and the trap community string. The default is **/etc/snmpd.conf**.

--trapd_conf_file= *filename*

The trap configuration file. It contains the Regular expression to Trap OID mappings. It is explained in the following section *Syntax of snmptrapd.conf file* below. It defaults to **/etc/snmptrapd.conf**.

--trap_program= *program*

This program actually launches the trap. It should be the snmptrapd program provided with the **ucd-snmp** agent. It defaults to **/sbin/snmptrapd**. This option is for debugging purposes.

Example

To send out an **Unknown Error** trap when a syslog line begins with **Error** or **error**, use the following line:

```
.1.3.6.1.4.1.3375.1.1.110.2.1(^[Ee]rror)Unknown  
Error
```

Files

/etc/snmptrapd.conf

/etc/syslogd.conf

/etc/snmpd.conf

config_httpd

The **config_httpd** tool is a curses-based configuration tool for the standard webserver. It requires fully-qualified domain names for both the internal and external interfaces by which the webserver will be known. It requires the user name and password for an administrative user, and asks if Support should be given access to the system.

If a user by that name already exists, it will request confirmation that you wish to change that user's password.

The **config_httpd** tool will then write the webserver configuration and authorization files.

Use the following syntax with **config_httpd**:

config_httpd

Caveats

Re-running the **config_httpd** tool again and changing the name of the administrative user will result in the prior user's access to the Configuration utility being dropped to read-only status.

config_httpsd

The **config_httpsd** tool is a curses-based configuration tool for the secure webserver. It requires fully-qualified domain names for both the internal and external interfaces by which the webserver will be known. It requires information to make a self-signed certificate for the webserver. It also requires the user name and password for an administrative user, and asks if Support should be given access to the system.

If a user by that name already exists, it requests confirmation that you wish to change that user's password.

The **config_httpsd** tool then writes the webserver configuration file and runs the SSL key generation facilities. This may take a minute or more on an average system.

Use the following syntax with **config_httpsd**:

config_httpsd

Caveats

Re-running the **config_httpsd** tool again and changing the name of the administrative user will result in the prior user's access to the Configuration utility being dropped to read-only status.

config_namesurfer

The **config_namesurfer** tool is a curses-based configuration tool for NameSurfer. It requires fully-qualified domain names that are known by the NameSurfer webserver. It also requires the user name and password for an administrative user.

The **config_namesurfer** tool initializes the NameSurfer databases and optionally calls **bind2namesurfer** to load master zone files listed in **/etc/named.conf** from BIND into NameSurfer, converts **/etc/named.conf** to use the new NameSurfer zone files, and writes the output of **bind2namesurfer** to **/var/3dns/etc/bind2namesurfer.log**.

Use the following syntax with **config_namesurfer**:

config_namesurfer

Caveats

You should run this program only once, as it attempts to initial the NameSurfer binary database files. However, running it multiple times has no effect.

config_rshd

The **config_rshd** tool is a Curses-based configuration tool for the rsh (remote shell) server. It requests a configuration address from which administrators may access the server. The address may contain wildcards. It also asks if the address for support may be included in the list of permitted hosts. If **inetd** is not currently configured, configure **inetd** for the requested services. If the service port for rsh is closed, it opens the service port to permit external access via rsh.

Use the following syntax with **config_rshd**:

config_rshd

Caveats

Re-running the **config_rshd** tool again and saying **no** will not result in rsh being deactivated. Also, administration addresses are appended, not replaced, with each re-run of **config_rshd**.

config_sshd

The **config_sshd** tool (configure ssh for BIG/ip) is a curses-based configuration tool for the ssh (secure shell) server. It requests a configuration address from which administrators may access the server via ssh. The address may contain wildcards. It also asks if the address for support may be included in the list of permitted hosts. If the service port for ssh is closed, it opens the service port to permit external access via ssh.

Use the following syntax with **config_sshd**:

config_sshd

Caveats

Re-running **config_sshd** again and saying **no** will not result in ssh being deactivated. Also, administration addresses are appended, not replaced, with each re-run of **config_sshd**.

syncd

The **syncd** daemon will synchronize a set of files between the 3DNS Controllers in its sync group. A **syncd** process is run independently on each controller and each **syncd** is responsible for detecting the modification of a local file and sending it to the remote **syncds** as well as receiving newly updated files from remote **syncds**.

The controllers in a sync group are defined by the **sync_group** statement in the **wideip.conf** file. The order of the controllers in the sync group in the **wideip.conf** file determine the priority of the

controllers to see which controller is a Principal or Receiver, but for **syncd** the order does not matter. Every controller in a sync group is a peer of each other.

Syncd's configuration files that list the files synchronized by **syncd** are listed in the files **sync_list** , **sync_list.mandatory** and **sync_list.named** (all in the directory **/var/3dns/etc**).

The files listed to be synchronized can be plain files, links (**syncd** will synchronize both the link and the actual file it points to), and directories (the directory and its contents, including sub-directories, will be synchronized). If a file in a synchronized directory is deleted, the deletion will be synchronized and the file will be removed from the same directory on all controllers.

The following files and directories are not allowed to be synchronized: **/** , **/usr**, **/etc**, **/var** (individual files or directories under those directories can be synchronized), and also the file **/etc/netstart**. Any file with the following suffixes will also never be synchronized: **.core** , **.core.gz** , **.bak** , **.no_sync**.

The files to synchronize are listed one per line in one the following **sync_list** files: **sync_list**, **sync_list.mandatory** and **sync_list.named** (all in the directory **/var/3dns/etc**). Each file may have one of the following tokens listed in front of the file name:

backup -

Each new version of this file is backed up whether changed locally or remotely. Directories are not backed up, but the files in the directory are backed up. Links are not backed up, but the file to which the link points is backed up. The backups are stored in the directory **/var/3dns/staging/backup**.

ckpoint -

The file will be checkpointed if **syncd** is requested to checkpoint its files (by default, any file with **backup** will be checkpointed). See the section ahead, titled *Checkpointing*.

fixed -

Once the file's **syncd** configuration is read in from a sync list, it cannot change, even if the sync list is reread in.

once -

The action associated with this file will be run only once, the first time the file is synchronized to the controller. See the following section, titled *Actions*.

no_sync -

The file will be watched, but never synchronized to other controllers.

Use the following syntax with **syncd**:

syncd [-f] | [-E] | [-s]

The options for **syncd** are:

-f

Runs **syncd** in the foreground (normally runs as a daemon).

-E

Prints a lot of debugging messages to **stderr** while running.

-s

Starts in standby 3DNS mode. A 3DNS Controller started in standby 3DNS mode may switch to active mode after startup. Standby 3DNS mode means it receives changes to synchronized files, but does not synchronize local changes to remote 3DNS Controllers.

Actions

An action is an executable program or script that is run by **syncd** whenever a file is synchronized from a remote controller. Arguments for an action can be listed after the action, and if one of the arguments for the action is to be the actual file name itself, you can use the **\$path** as the argument. Actions are run on files, not on directories (but on the individual synchronized files in the directory), and not on links (but on the file that the link points to). Each line of a sync list file consist of:

```
[token1|token2|...|tokenN] file_name [action]  
[action args]
```

Comments can be put in the sync list files, starting with #.

The sync list, **/var/3dns/etc/sync_list**, by default is empty. Also, no actions are allowed for any files listed in this sync list. This file is synchronized between controllers (meaning, if you add a new file to **sync_list**) the new file and **sync_list** will be synchronized to the remote controllers.

Here is an example of what you could put into **sync_list** :

```
# A config file that is backed up  
backup /etc/my_config  
# A directory with things that I want to sync and  
# be able to checkpoint  
ckpoint /var/my_stuff
```

The sync list **/var/3dns/etc/sync_list.mandatory** are the files that are synchronized for managing the 3DNS Controller configuration. The list should rarely require modification:

```
# This lists files that syncd will always sync  
# between 3DNS Systems and the actions that will  
# be run on the remote 3DNS systems:  
fixed backup /var/3dns/etc/wideip.conf  
3dns_load $path  
  
# Synced include files directory  
fixed backup /var/3dns/include/sync/
```

```
# The extra commands files that may be produced
# when a collector dumps (this is a directory).
# The action is only run once the file first
# appears
fixed once /var/run/sync_wideip_cmds/
3dns_action $path

# Namesurfer stuff
fixed backup /var/f5/namesurfer/db/ ns_restart
$path

# Uncomment the next line if /etc/named.conf
# should also sync
#fixed backup /etc/named.conf 3ndc
restart
```

The **sync_list.named** file is created when **named** starts up. It lists DNS zone files that need to be synchronized (if any). This file should not be modified by the user.

Checkpointing

A checkpoint is a saved version of the files that **syncd** synchronizes at a certain moment. Only files that have the **backup** or **ckpoint** token are checkpointed. If you later need to restore the state of a controllers configuration files, the checkpoint can be rolled back to restore the previous state of the controller. See the manual pages for **syncd_checkpoint** and **syncd_rollback** to see how to create and rollback a checkpoint.

Scripts

This section provides information on each script that is shipped with the 3DNS Controller. Many scripts correspond to commands on the 3DNS Maintenance menu, which is discussed starting on page 7 -1. This section provides information about how the scripts work. If you plan on doing a scripted task manually, you should find this section especially helpful.

◆ Note

Before you edit a script, make a backup copy of the original.

3dns_admin_start

The **3dns_admin_start** script corresponds to the **Restart 3DNS Web Administration** command on the 3DNS Maintenance menu. This command restarts the 3DNS web server.

3dns_auth

All 3DNS Controller scripts are easier to use when you generate password authentication. The **3dns_auth** script corresponds to the **Generate RSA Authentication** command on the 3DNS Maintenance menu.

◆ Note

This script is not available in the international version of the 3DNS Controller.

The **3dns_auth** script generates a password authentication copying the **ssh** key to each 3DNS Controller and BIG/ip Controller.

◆ WARNING

*Before you use this command, you must set the **RSAAuthentication** parameter to **yes** in the **/etc/sshd_config.conf** file.*

If no **identity.pub** file exists, **3dns_auth** runs the **ssh-keygen** command to generate **/root/.ssh/identity** and **/root/.ssh/identity.pub** files that incorporate NULL passphrases. An existing **identity.pub** file indicates that **ssh-keygen** was already run. Running **ssh-keygen** more than once will cause problems, and is not recommended.

When you run **ssh-keygen**, press Enter when asked for a passphrase. Do not type in a password.

Here is a sample session to generate a public key:

```
3dns-receiver# ssh-keygen
Initializing random number generator...
Generating p: .....++ (distance 364)
Generating q: ..++ (distance 16)
Computing the keys...
Testing the keys...
Key generation complete.
Enter file in which to save the key (/root/.ssh/identity):
Enter passphrase:
Enter the same passphrase again:
Your identification has been saved in /root/.ssh/identity.
Your public key has been saved in /root/.ssh/identity.pub
```

To test that you have successfully generated the **ssh** key, use **ssh** to log into the principal 3DNS Controller without a password:

```
principal# ssh root@<ip-address-of-3DNS>
```

If you have an existing **identity.pub** file

If you have an existing **identity.pub** file, but you want to perform the other tasks performed by **3dns_auth**, do not run the script again. Instead, append the contents of the **/root/.ssh/identity.pub** file to the 3DNS **/root/.ssh/authorized_keys** file, using the following command:

```
3dns-standby# cat /root/.ssh/identity.pub |\
ssh -l root <ip-address-of-3DNS> 'cat >>
/root/.ssh/authorized_keys'
```

Note that you must use a front tick mark (also called a single straight quotation mark) in the above syntax.

3dns_dump

The **3dns_dump** script saves the current state of the **named** cache into a new **/var/3nds/etc/wideip.conf** file.

3dns_sync_metrics

The **3dns_sync_metrics** script corresponds to the **Synchronize Metrics Data** command on the 3DNS Maintenance menu. You should only use this script when you are configuring a new 3DNS Controller. This script prompts you to either copy metrics data from the local 3DNS Controller to other 3DNS Controllers, or from a remote 3DNS Controller to the local 3DNS Controller.

3dns_web_config

The **3dns_web_config** script corresponds to the **Reconfigure 3DNS Web Administration** command on the 3DNS Maintenance menu. This script lets you make configuration changes to the 3DNS web server.

3dns_web_passwd

The **3dns_web_passwd** script corresponds to the **Change/Add Users for 3DNS Web Administration** command on the 3DNS Maintenance menu. This script secures the 3DNS web server using basic authentication. This script lets you provide restricted or administrative access to the 3DNS web server for selected users

only, and assigns passwords for those users. Users with restricted access have access to the statistics area only. Users with administrative access have access to all areas of the 3DNS web server.

◆ Note

---

*The **3dns_web_passwd** script is run by the First-Time Boot utility. You can run this script again any time you need to provide access for another user.*

3dnsmaint

The **3dnsmaint** script opens the 3DNS Maintenance menu.

3dprint

The **3dprint** script corresponds to the **Dump and List named Database** command on the 3DNS Maintenance Menu. This script lets you view these statistics screens on the command line:

❖ **sum**

Displays summary statistics, such as the 3DNS Controller version, the total number of resolved requests, and the load balancing methods used to resolve requests.

❖ **paths**

Displays path statistics, such as round trip time and packet completion rate.

❖ **ldns**

Displays statistics collected for local DNS servers, including the number of resolution requests received from a given server, and the current protocol used to probe the server.

❖ **vs**

Displays statistics about BIG/ip and host virtual servers, such as the server state, and the number of times it has received resolution requests.

❖ **bigips**

Displays statistics about all BIG/ip Controllers known to the 3DNS Controller, including the number of virtual servers each BIG/ip Controller manages, and the number of times that the 3DNS Controller resolves requests to those virtual servers.

❖ **hosts**

Displays statistics about all hosts known to the 3DNS Controller, including the number of times that the 3DNS Controller resolves requests to the host.

❖ **wips**

Displays statistics about each wide IP defined on the 3DNS Controller, including load balancing information and the remaining time to live before the wide IP's metrics data needs to be refreshed.

3ndc

The **3ndc** script starts the **3ndc** utility, which is described on page B -4. **ndc** is an alias for **3ndc**.

big3d_check

The **big3d_check** script corresponds to the **Check big3d** command on the 3DNS Maintenance menu. This script checks that each BIG/ip Controller listed in the **bigips.txt** file is running the **big3d** agent.

big3d_install

The **big3d_install** script corresponds to the **Install and Start big3d** command on the 3DNS Maintenance menu. This script installs and starts the appropriate version of the **big3d** agent on each BIG/ip Controller. This script is useful for 3DNS Controller updates.

big3d_install performs the following procedure on each BIG/ip Controller:

1. Stops the running **big3d** agent process.
2. Uses a matrix file to determine which version of the **big3d** agent to copy to the BIG/ip Controller. The matrix file is a file that lists version numbers for all BIG/ip Controllers known to the 3DNS Controller and the version numbers of the **big3d** agent and **named** utility running on each BIG/ip Controller.
3. Adds the following to the bottom of the **/etc/rc.local** file:

```
if [ -f /usr/sbin/big3d ]; then
    echo -n "big3d": /usr/sbin/big3d 2>
/dev/null
fi
```
4. Starts **/usr/sbin/big3d**.

For configuration options, see *big3d* on page B-14.

big3d_restart

The **big3d_restart** script corresponds to the **Restart big3d** command on the 3DNS Maintenance menu. This script stops and restarts the **big3d** agent on each BIG/ip Controller.

big3d_version

The **big3d_version** script corresponds to the **Check versions of named, BIG/ip kernel and needed big3d** command on the 3DNS Maintenance menu. This script displays version numbers for all BIG/ip Controllers known to the 3DNS Controller, as well as the version numbers of the **big3d** agent and **named** utility running on each BIG/ip Controller.

edit_lock

The **edit_lock** script lets you safely edit a specified file that is synchronized between 3DNS Controllers in a sync group. This script creates a temporary version of the original file, and this temporary file replaces the original file when you are finished editing it. If you do not use this script to edit a file, there is the danger that a partial file might be synchronized to other 3DNS Controllers in the sync group.

To use this script, type the following:

edit_lock <file name>

edit_wideip

The **edit_wideip** script corresponds to the **Edit 3DNS Configuration** command on the 3DNS Maintenance menu. This script opens the **wideip.conf** file for editing, copies it to all other 3DNS Controllers in the local 3DNS Controller's sync group, and restarts **named**.

install_key and F5makekey

The **install_key** script corresponds to the **Generate and Copy F5 iQuery Encryption Key** command on the 3DNS Maintenance menu. This script starts the **F5makekey** script and generates a seed key for encrypting communications between the 3DNS Controller and BIG/ip Controller. The **install_key** script creates and distributes the iQuery key to all BIG/ip Controllers and other 3DNS Controllers on your network.

◆ Note

This script is not available in the international version of 3DNS Controller.

To start the **F5makekey** script, type the following from **/usr/contrib/bin**:

f5makekey

The seed value is located in **/etc/F5key.dat** and contains a random length (12-52) of random content (1-255), created by **F5makekey**. This array of values is used by MD-160, a one-way hash function, to generate a key (7 characters in length) for the Blowfish encryption algorithm.

syncd_checkpoint

The **syncd_checkpoint** script corresponds to the **Checkpoint synced files** command on the 3DNS Maintenance menu. This script creates a **checkpoint file**. A checkpoint file is a compressed tar file that contains an archive of the files that are synchronized.

You can run this script with or without arguments. If you run **syncd_checkpoint** without specifying arguments, the script creates the following default checkpoint file:

/var/3dns/staging/checkpoint/default.tar.gz

Note

*All checkpoint file names have a **.tar.gz** suffix.*

The **syncd_checkpoint** script can take the following optional arguments:

syncd_checkpoint [-c <name>] [-i]

The options for **syncd_checkpoint** are defined as follows:

-c <name>

Creates a checkpoint file with the specified file name. You can also specify a non-default path for the file, unless the path starts with a slash (/). The default path for checkpoint files is **/var/3dns/staging/checkpoint/**. The **syncd_checkpoint** script automatically appends a **.tar.gz** extension to the end of the file name.

-i

Runs the script in an interactive session, which means that you are prompted for a file name.

syncd_rollback

The **syncd_rollback** script corresponds to the **Rollback checkpoint** command on the 3DNS Maintenance menu. This script unrolls a checkpoint file, which contains an archive of all synchronized files. This has the effect of replacing the current files with the files archived in the checkpoint file.

The **syncd_rollback** script can take the following optional arguments:

syncd_rollback [-c] [-c <name>] [-r] [-u] [-i]

The options for **syncd_rollback** are defined as follows:

-c

Unrolls the most recently created checkpoint file, whether it is in the default location or elsewhere.

-c <name>

Unrolls the specified checkpoint file, whether it is in the default location or elsewhere. It is not necessary to end the name with **.tar.gz**, as this suffix is assumed.

-r

The archived files are restored with their old timestamps. This means that if any of the synchronized files were updated on a remote 3DNS Controller, the updated files will overwrite any older files contained in the checkpoint file.

-u

The archived files are restored with updated timestamps with the current time. This means that the files in the checkpoint are synchronized to the remote 3DNS Controllers and overwrite the existing files on the remote 3DNS Controllers.

-i

Runs the script in an interactive session, which means that you are prompted for option information.

◆ Note

*When you run this script from the command line, you must use the **-r**, **-u**, or **-i** option.*

syncd_start

The **syncd_start** script corresponds to the **Restart syncd** command on the 3DNS Maintenance menu. This script restarts the **syncd** daemon if it is already running, or starts it if it is not.

You can run this script with or without arguments. If you run **syncd_start** without specifying arguments, the script starts or restarts **syncd**.

The **syncd_start** script can take the following optional arguments:

syncd_start [-c] [-c <name>] [-r] [-u] [-i]

The options for **syncd_start** are defined as follows:

-c

Before restarting **syncd**, unrolls the most recently created checkpoint file, whether it is in the default location or elsewhere.

-c <name>

Before restarting **syncd**, unrolls the specified checkpoint file, whether it is in the default location or elsewhere. It is not necessary to end the name with **.tar.gz**, as this suffix is assumed.

-r

Restores the archived files with their old timestamps. This means that if any of the synchronized files were updated on a remote 3DNS Controller, the updated files overwrite the rolled back files.

-u

Restores the archived files with updated timestamps to the current time. This means that the files in the checkpoint file overwrite any updated files on remote 3DNS Controllers.

-i

Runs the script in an interactive session, which means that you are prompted for option information.

◆ Note

*When you use the **-c** option, you must also use either the **-r** or **-u** option.*

syncd_stop

The **syncd_stop** script corresponds to the **Stop syncd** command on the 3DNS Maintenance menu. This script stops the **syncd** daemon, if it is running.

You can run this script with or without arguments. If you run **syncd_stop** without specifying arguments, the script simply stops **syncd**.

The **syncd_stop** script can take the following optional arguments:

syncd_stop [-c] [-c <name>] [-i]

The options for **syncd_stop** are defined as follows:

-c

Creates a checkpoint file in the default location before stopping **syncd**.

-c name

Creates a checkpoint file with the specified name and path before stopping **syncd**.

-i

Runs the script in an interactive session, which means that you are prompted for option information.



C

BIND 8 Information



BIND 8 overview

Although you can use earlier versions of BIND (version 4.97 and later), F5 Networks recommends that you use BIND 8.1.2 or later with the 3DNS Controller.

For more information on BIND, refer to the Internet Software Consortium Web site at **www.isc.org**.

BIND 8 has the advantage of being more configurable than earlier versions of BIND. New areas of configuration, such as access control lists (ACLs) and categorized logging, are now available. You can selectively apply more options, rather than being required to apply options to all zones. To incorporate this new technology and provide for future enhancements, BIND 8 requires a new format for configuration files.

A BIND 8 configuration file consists of two types of information: statements and comments. Both of these are described in the following sections. This appendix also describes the relationship between BIND and 3DNS Controller wide IP definitions.

Statements

BIND statements end with a semicolon. Statements can contain blocks of sub-statements, which are also terminated with a semicolon.

The following statements are supported:

Statement	Description
acl	Defines a named IP address matching list, for access control and other uses.
include	Includes a file.
key	Specifies key information for use in authentication and authorization.
logging	Specifies what the server logs, and where the log messages are sent. This statement may only be used once per configuration.
options	Controls global server configuration options and sets defaults for other statements. This statement may only be used once per configuration.
server	Sets certain configuration options on a per-server basis.
zone	Defines a zone.

Table C.1 BIND statements

acl statement

The **acl** statement creates a named address match list. It gets its name from a primary use of address match lists: Access Control Lists (ACLs).

Note that an address match list's name must be defined with **acl** before it can be used elsewhere; no forward references are allowed.

The following ACLs are built in:

ACL	Description
any	Allows all hosts.
none	Denies all hosts.
localhost	Allows the IP addresses of all interfaces on the system.
localnets	Allows any host on a network for which the system has an interface.

Table C.2 Supported ACLs

Syntax

```
acl <name> {  
    address_match_list  
};
```

key statement

The **key** statement defines a key ID which can be used in a server statement to associate an authentication method with a particular name server.

The **key** statement is intended for future use by the server. It is checked for syntax but is otherwise ignored.

Syntax

```
key <key_id>{  
    algorithm <algorithm_id>;  
    secret <secret_string>;  
};
```

logging statement

The **logging** statement configures a wide variety of logging options for the name server.

Syntax

```
logging {
  [ channel <channel_name> {
    ( file <path_name>
      [ versions ( number | unlimited ) ]
      [ size <size_spec> ]
    | syslog ( kern | user | mail | daemon |
      auth | syslog | lpr |
      news | uucp | cron | authpriv | ftp |
      local0 | local1 | local2 | local3 |
      local4 | local5 | local6 | local7 )
    | null );
    [ severity ( critical | error | warning | notice
    |
      info | debug [ level ] | dynamic ); ]
    [ print-category <yes | no>; ]
    [ print-severity <yes | no>; ]
    [ print-time <yes | no>; ]
  }; ]
  [ category <category_name> {
    <channel_name>; [ <channel_name>; ... ]
  }; ]
  ...
};
```

options statement

The **options** statement sets up global options to be used by BIND. This statement should appear only once in a configuration file; if BIND finds more than one occurrence, BIND honors the first. When this happens, BIND generates a warning alerting you that your configuration contains multiple **options** statements. If BIND does not find an **options** statement in the configuration file, BIND uses an **options** block, with each option set to its default.

Syntax

```
options {
    [ directory <path_name>; ]
    [ named-xfer <path_name>; ]
    [ dump-file <path_name>; ]
    [ memstatistics-file <path_name>; ]
    [ pid-file <path_name>; ]
    [ statistics-file <path_name>; ]
    [ auth-nxdomain <yes | no>; ]
    [ deallocate-on-exit <yes | no>; ]
    [ fake-iquery <yes | no>; ]
    [ fetch-glue <yes | no>; ]
    [ host-statistics <yes | no>; ]
    [ multiple-cnames <yes | no>; ]
    [ notify <yes | no>; ]
    [ recursion <yes | no>; ]
    [ forward ( only | first ); ]
    [ forwarders { [<in_addr>;<in_addr>;...]]}; ]
    [ check-names (master | slave | response )
      ( warn | fail | ignore); ]
    [ allow-query { <address_match_list> }; ]
    [ allow-transfer { <address_match_list> }; ]
    [ listen-on [ port <ip_port> ]
      { <address_match_list> }; ]
```

```
[ query-source [ address ( <ip_addr> | * ) ]
[ port ( <ip_port> | * ) ] ; ]
[ max-transfer-time-in <number>; ]
[ transfer-format (one-answer|many-answers);]
[ transfers-in <number>; ]
[ transfers-out <number>; ]
[ transfers-per-ns <number>; ]
[ coresize <size_spec> ; ]
[ datasize <size_spec> ; ]
[ files <size_spec> ; ]
[ stacksize <size_spec> ; ]
[ cleaning-interval <number>; ]
[ interface-interval <number>; ]
[ statistics-interval <number>; ]
[ topology { <address_match_list> }; ]
};
```

server statement

The **server** statement defines the characteristics associated with a remote name server.

Syntax

```
server <ip_addr> {
  [ bogus <yes | no>; ]
  [ transfers <number>; ]
  [ transfer-format (one-answer|many-answers);]
  [ keys { <key_id> [key_id ... ] }]; ]
};
```

zone statement

The **zone** statement defines a zone.

Syntax

```
zone <domain_name> [ ( in|hs|hesiod|chaos )]{
    type master;
    file <path_name>;
    [ check-names ( warn | fail | ignore ); ]
    [ allow-update { <address_match_list> }; ]
    [ allow-query { <address_match_list> }; ]
    [ allow-transfer { <address_match_list> }; ]
    [ notify <yes | no>; ]
    [ also-notify { <ip_addr>; [ <ip_addr>;...]}];
};

zone <domain_name> [ ( in|hs|hesiod|chaos) ] {
    type ( slave | stub );
    [ file <path_name>; ]
    masters { <ip_addr>; [ <ip_addr>; ... ] };
    [ check-names ( warn | fail | ignore ); ]
    [ allow-update { <address_match_list> }; ]
    [ allow-query { <address_match_list> }; ]
    [ allow-transfer { <address_match_list> }; ]
    [ max-transfer-time-in <number>; ]
    [ notify <yes | no>; ]
    [ also-notify { <ip_addr>; [ <ip_addr>;...]}];
};

zone "." [ ( in | hs | hesiod | chaos ) ] {
    type hint;
    file <path_name>;
    [ check-names ( warn | fail | ignore ); ]
};
```

Comments

BIND 8 comments follow syntax rules that are similar to the 3DNS Controller comments syntax rules.

You can insert comments anywhere you would otherwise see white space in a BIND configuration file.

Syntax

Note that the comment syntax depends on the environment in which you use the configuration file. For example:

```
/* This is a BIND comment as in C */  
// This is a BIND comment as in C++  
# This is a BIND comment as in common Unix shells  
  and Perl
```

Definition and usage

The format for comments varies by programming language; each format is described below.

C style comments

C style comments start with the slash character, followed by the asterisk character (/*), and end with the asterisk character, followed with the slash character (*). Because the comment is completely delimited with these characters, a comment can span multiple lines.

Note that C style comments cannot be nested. For example, the following is not valid because the entire comment ends with the first */:

```
/* This is the start of a comment.  
   This is still part of the comment.  
/* This is an incorrect attempt to nest a comment. */  
   This is no longer in any comment. */
```

C++ style comments

C++ style comments start with two slash characters (//) and are no longer than one line in length. To have one logical comment span multiple lines, each line must start with the // pair.

For example:

```
// This is the start of a comment. The next line
// is a new comment line, even though it is
// logically part of the previous comment.
```

Shell style comments

Shell style (also known as Perl style) comments start with the # character and are no longer than one line in length.

For example:

```
# This is the start of a comment. The next line
# is a new comment line, even though it is logically
# part of the previous comment.
```

WARNING

You cannot use the semicolon (;) character to start a comment as you would in a zone file. The semicolon indicates the end of a configuration statement. Text following a semicolon is interpreted as the start of the next statement.

Converting older configuration files to BIND 8 format

You can convert BIND 4.9.x configuration files to the BIND 8 format using **/etc/named-bootconf.pl**, a Perl script that is part of the BIND 8.1 source kit.

Relating BIND information to 3DNS Controller wide IP definitions

We recommend that you use NameSurfer, which is included with the 3DNS Controller, to configure BIND. However, if you want to manually configure BIND and understand how BIND relates to wide IP definitions, read on. This section provides an example procedure for adding a wide IP when BIND is manually configured.

◆ Note

*The following information assumes you have read O'Reilly & Associates' book **DNS and BIND** (third edition). You can purchase this book from a technical bookstore.*

Before defining a wide IP

Before configuring a wide IP, gather your 3DNS Controller, BIG/ip Controller, host, and virtual server information.

The following information includes sample wide IP statements that derive from the sample configuration file that starts on page A -50. The sample wide IP statement configures a wide IP for the **domain.com** domain, where the IP addresses assigned to the 3DNS Controller interfaces are shown in the table below.

3DNS Controller	IP address
New York	192.168.101.2
Los Angeles	192.168.102.2

Table C.3 3DNS Controller addresses

Gather your BIG/ip Controller and host configuration information so that you can easily see which virtual servers have the replicated content.

For example, create tables like the following. In the first table, list each server.

Server	IP address	Server type
New York	192.168.101.40	BIG/ip Controller
Los Angeles	192.168.102.40	BIG/ip Controller
Tokyo	192.168.103.40	BIG/ip Controller
Tokyo	192.168.104.40	Host
Los Angeles	192.168.105.40	Host

Table C.4 Server addresses

Next, create a table that lists the virtual servers managed by each BIG/ip Controller (include only those that host content for the domain you are load balancing). For example, each virtual server

in the following table is owned by a different BIG/ip Controller, yet each contains identical content.

BIG/ip Controller	Virtual server	Virtual port
New York	192.168.101.50	80
Los Angeles	192.168.102.50	80
Tokyo	192.168.103.50	80

Table C.5 Virtual servers owned by BIG/ip Controllers

You configure virtual servers as part of the BIG/ip Controller configuration process. See the **BIG/ip Controller Administrator Guide**. List the other host machines and the IP addresses of the virtual servers that contain the same content.

Host	Virtual server	Virtual port
Tokyo	192.168.104.50	80
Los Angeles	192.168.105.50	80

Table C.6 Virtual servers owned by host machines

Defining a wide IP

The following procedure shows how to manually add a wide IP.

To add a wide IP and manually configure BIND

1. Find or create the top level domain configuration file. This file is usually found in the **/etc** directory.
  - For BIND 4, enter the following line in the **named.boot** file:
primary domain.com db.domain.com
  - For BIND 8, enter the following in the **named.conf** file:

```
zone "domain.com" IN {  
    type master;  
    file "db.domain.com";  
};
```

To specify a type other than **master**, see the syntax for the **zone** statement on page C -7.

2. If your network's master DNS is not a 3DNS Controller, create a new subdomain to be controlled by the 3DNS Controller.

For example, to create a subdomain called **wip.domain.com**, do one of the following:

- Add the new subdomain to the **named.conf** file with the following lines:

```
zone "wip.domain.com" IN {  
    type master;  
    file "db.wip.domain.com";  
};
```

- If the 3DNS Controller does not manage the top level domain, the subdomain must be delegated to each 3DNS Controller on your network. To delegate the domain to each 3DNS Controller in your network, add lines like the following to the top level domain database file (**db.domain.com** in this example):

```
wip IN NS 3dns.newyork  
    IN NS 3dns.losangeles  
3dns.newyork IN A 192.168.101.2  
3dns.losangeles IN A 192.168.102.2
```

3. If your network's master DNS is not a 3DNS Controller, change (or add) the target domain name to an alias.

For example, you might find the target domain as an **A** record in your name server's DNS database as follows:

```
www    IN    A    192.168.101.50
```

Edit **db.domain.com** so that it contains following line:

```
www    IN    CNAME    www.wip
```

In the above line, **www.wip.domain.com** is the domain name controlled by the 3DNS Controller.

4. Next, you need to choose a wide IP key. Select one of the virtual servers in the group, and use its IP address as the wide IP key. In this example, **192.168.101.50** is the wide IP key for **www.wip.domain.com**.
5. Configure the load balanced name on the 3DNS Controller.

Locate or create a subdomain database file for **wip.domain.com**. Select one IP address from the set and add an **A** record for the **www.wip** domain. Use the IP address as the wide IP key. In the new **A** record, specify a low TTL value. (You can override the database's global TTL value for an individual name.)

The following is an example of an entire zone file. The next to last line is the **A** record:

```

wip.domain.com. IN SOA 3dns.newyork.domain.com.
postmaster.domain.com. (
                        1998062914 ; Serial as YYYYMMDDXX
                        3600 ; Refresh
                        900 ; Retry
                        3600000 ; Expire
                        2 ) ; Minimum (default ttl for entire file)
; Domain DNS servers
wip.domain.com. IN NS 3dns.newyork.domain.com.
                  IN NS 3dns.losangeles.domain.com.
; Glue records
3dns.newyork.domain.com. IN A 192.168.101.2
3dns.losangeles.domain.com. IN A 192.168.102.2
; Mail servers
domain.com. IN MX 10 mx.newyork.domain.com.
domain.com. IN MX 20 mx.losangeles.domain.com.
; Regular Host
otherbox IN A 192.168.101.20
; domain name TTL Wide IP key
www 1 IN A 192.168.101.50
ftp IN A 192.168.101.60

```

Figure C.1 Sample zone file for *wip.domain.com*.

The following example is provided for reference only. If you need help establishing reverse domains (address-to-name mappings), refer to the ***DNS and BIND*** book mentioned at the start of this procedure. The following sample screens show the reverse domain mapping files on the New York 3DNS Controller:

```
101.168.192.in-addr.arpa. IN SOA 3dns.newyork.domain.com.  
postmaster.domain.com. (  
    1998062914 ; Serial as YYYYMMDDXX  
    3600 ; Refresh  
    900 ; Retry  
    3600000 ; Expire  
    14000 ) ; Minimum  
  
101.168.192.in-addr.arpa. IN NS 3dns.newyork.domain.com.  
                           IN NS 3dns.losangeles.domain.com.  
  
20          IN PTR otherbox.wip.domain.com.  
50          IN PTR www.wip.domain.com.  
60          IN PTR ftp.wip.domain.com.
```

Figure C.2 Excerpt from **db.192.168.101**

◆ **Note**

Because a virtual server is listed in each data center for a wide IP definition, you need to define an entry to mapping for each class C network that is included in the wide IP definition.

```

102.168.192.in-addr.arpa. IN SOA 3dns.newyork.domain.com.
postmaster.domain.com. (
                        1998062914 ; Serial as YYYYMMDDXX
                        3600 ; Refresh
                        900 ; Retry
                        3600000 ; Expire
                        14000 ) ; Minimum

102.168.192.in-addr.arpa. IN NS 3dns.newyork.domain.com.
                        IN NS 3dns.losangeles.domain.com.

50   IN PTR www.wip.domain.com.
60   IN PTR ftp.wip.domain.com.

```

Figure C.3 Excerpt from **db.192.168.102**

```

103.168.192.in-addr.arpa. IN SOA 3dns.newyork.domain.com.
postmaster.domain.com. (
                        1998062914 ; Serial as YYYYMMDDXX
                        3600 ; Refresh
                        900 ; Retry
                        3600000 ; Expire
                        14000 ) ; Minimum

103.168.192.in-addr.arpa. IN NS 3dns.newyork.domain.com.
                        IN NS 3dns.losangeles.domain.com.

50   IN PTR www.wip.domain.com.
60   IN PTR ftp.wip.domain.com.

```

Figure C.4 Excerpt from **db.192.168.103**

Instead of a typical one-to-one relationship, where one address maps to one name, the following addresses all map to **www.wip**:

192.168.101.50

192.168.102.50

192.168.103.50

6. Add the **www.wip.domain.com** domain as a wide IP to your **wideip.conf** file. You can define which load balancing mode you want to use for the wide IP now or after you read Chapters 5 and 6. List which virtual servers are to be available for load balancing this wide IP.

Here is an example of a **wideip** statement to add to **wideip.conf**:

```
wideip {
    address      192.168.101.50
    service      "http"
    name         "www.wip.domain.com"
    ttl          60      // increase the domain default ttl
    pool {
        name     "Pool_1"
        ratio     2      // applies to pool_lbmode == ratio
        preferred leastconn
        alternate ratio
        address   192.168.101.50    ratio 2
        address   192.168.102.50    ratio 1
        address   192.168.103.50    ratio 1
    }
    pool {
        name     "Pool_2"
        type      VSb
        ratio     1
        preferred rr
        address   192.168.102.60    ratio 2
        address   192.168.103.60    ratio 1
    }
}
```

Figure C.5 Sample wideip statement

The wide IP is now in place and configured.

To add additional wide IPs by manually editing the configuration file

The following example describes how to add a wide IP named **ftp.wip.domain.com**:

1. Select a set of geographically distributed virtual servers.
2. Select the IP address of one of the virtual servers in the set to be the wide IP key (see *Understanding the wideip key*, on page 2-25 for details).
3. Define the wide IP name and key within BIND by adding the following resource record to **db.wip.domain.com**:
ftp.wip IN A 192.168.102.60
4. Define the virtual server list and the wide IP key within the 3DNS Controller by adding it to the **wideip.conf** file as follows:

```
wideip {
    address      192.168.102.60
    service      "ftp"
    name         "ftp.wip.domain.com"
    pool {
        name     "main_pool"
        type      VSb
        preferred leastconn
        alternate ratio
        address   192.168.101.60    ratio 2
        address   192.168.102.60    ratio 4
        address   192.168.103.60
    }
}
```

Figure C.6 Sample *wideip* statement

5. Restart the 3DNS Controller by typing the following:
ndc restart

Understanding zone minimums

The zone file contains a Minimum field in the SOA (Start of Authority) section of the file. The Minimum value is the TTL for all resource records (RR) in the zone file. However, you can override the zone minimum for a given RR.

For example, if you do not want a DNS to cache the previously issued answer for a domain name, you can specify a very low value for the Minimum field.

◆ Note

*For wide IP domain names, specify the TTL in the **wideip** statement. See **The wide IP statement**, on page A -39.*

In the following zone file excerpt, the specified Minimum value is 30 seconds for every entry. The exception is the domain name **www.wip**, which is overridden and not saved in any DNS cache. The result is that a new query is made each time there is a name resolution request for **www.wip**. This allows the 3DNS Controller to respond with the most intelligent answer for each request.

```
wip.domain.com. IN SOA 3dns.newyork.domain.com.  
postmaster.domain.com. (  
    1998062914 ; Serial as YYYYMMDDXX  
    3600 ; Refresh  
    900 ; Retry  
    3600000 ; Expire  
    30 ) ; Minimum (default ttl for entire file)  
www.wip      0      IN      A      192.168.101.60
```

Figure C.7 Zone minimums

Replacing your DNS servers with 3DNS Controllers as master DNS servers for your domain

If you are replacing your master DNS with 3DNS Controllers, the method of importing your old zone files is dependent on which of the options you chose during your planning stage in converting existing BIND files.

If you skipped NameSurfer configuration in the First-Time Boot utility

The **config_namesurfer** script will convert your old zone files and configuration file. The BIND files must be in place before you configure the NameSurfer application.

To transfer and convert existing BIND files

1. Manually transfer the zone files listed as master zones in **/etc/named.conf** on the existing BIND server to **/var/namedb/*** on the principal 3DNS Controller. On US 3DNS Controllers you can use **scp** to transfer files as well as **rnp** or **ftp** in passive mode. On international 3DNS Controllers you can use only **rnp** or **ftp** in passive mode.
2. On the 3DNS Controller run the **config_namesurfer** script using the command below, and responding **yes** when asked if you want NameSurfer to be the master.

config_namesurfer

After the script completes and the NameSurfer application starts, the application automatically converts the transferred zone files from BIND format to NameSurfer format and converts the **/etc/named.conf** file to use the NameSurfer zone files.

The next step is dependent on whether you are using the advanced synchronization features of 3DNS and intend to use multiple master DNSs.

If you intend to use multiple master DNSs without synchronization

Repeat steps 1 and 2 listed in the previous section, *To transfer and convert existing BIND files*, on each subsequent 3DNS Controller.

◆ Note

Any changes you make on one 3DNS Controller must be made on all the others.

If you intend to use multiple masters with synchronization

1. Run the First-Time Boot utility on the remaining 3DNS Controllers in the network, but be certain to choose **yes** when asked if you want NameSurfer to run as a master.
2. On all master 3DNS Controllers manually edit the `/var/3dns/etc/sync_list.mandatory` file and uncomment the line:
fixed backup /etc/named.conf 3ndc reload
3. Return to the principal 3DNS Controller and add the remaining 3DNS Controllers to the sync group.

Once you finish setting up all of the 3DNS Controllers and adding each of them to the sync group, the NameSurfer application automatically broadcasts the master zone files to each 3DNS Controller in the group. All changes that are made with Namesurfer are automatically broadcast to all other 3DNS Controllers in your sync group. In addition, zone files changes may be made on any of the 3DNS Controllers.

If you intend to use one primary 3DNS Controller with one or more secondary 3DNS Controllers without synchronization

Run the First-Time Boot utility on the remaining 3DNS Controllers in the network, but be certain to choose **no** when asked if you want NameSurfer to run as a master.

◆ Note

In this configuration you must perform all your zone edits only on the primary DNS server, which is now the 3DNS Controller.

If you intend to use one primary 3DNS Controller with one or more secondary 3DNS Controllers with synchronization

Run the First-Time Boot utility on the remaining 3DNS Controllers in the network, but be certain to choose **yes** when asked if you want NameSurfer to run as a master.

◆ **Note**

In this configuration you can perform zone edits on your secondary DNS server, and these changes will be broadcast to the primary DNS server. Normal BIND zone transfers will occur. If your primary DNS server becomes unavailable, your zone edits will not be broadcast.

Running 3DNS Controllers as DNS masters only for wide IP sub-domains

3DNS Controllers must be the authority for the zone files associated with wide IP definitions. When you set up a configuration where the 3DNS Controllers are DNS master only for those sub-domains, you need to make a few changes to the zone files on the master DNS for your domain. Refer to the section *Relating BIND information to 3DNS Controller wide IP definitions*.

The next step is dependent on whether you are using the advanced synchronization features of 3DNS and intend to use multiple master DNSs.

If you intend to use multiple masters without synchronization

Run the First-Time Boot utility on the remaining 3DNS Controllers in the network, but be certain to choose **yes** when asked if you want NameSurfer to run as a master.

◆ **Note**

Any changes you make on one 3DNS Controller must be made on all the others.

If you intend to use multiple masters with synchronization

1. Run the First-Time Boot utility on the remaining 3DNS Controllers in the network, but be certain to choose **yes** when asked if you want NameSurfer to run as a master.
2. On all master 3DNS Controllers manually edit the `/var/3dns/etc/sync_list.mandatory` file and uncomment the line:
fixed backup /etc/named.conf 3ndc reload
3. Return to the principal 3DNS Controller and add the remaining 3DNS Controllers to the sync group.

Once you finish setting up all of the 3DNS Controllers and adding each of them to the sync group, the NameSurfer application automatically broadcasts the master zone files to each 3DNS Controller in the group.

If you intend to use one primary 3DNS Controller with one or more secondary 3DNS Controllers without synchronization

Run the First-Time Boot utility on the remaining 3DNS Controllers in the network, but be certain to choose **no** when asked if you want NameSurfer to run as a master.

◆ Note

In this configuration you must perform all your zone edits only on the primary DNS server, which is now the 3DNS Controller.

If you intend to use one primary 3DNS Controller with one or more secondary 3DNS Controllers with synchronization

Run the First-Time Boot utility on the remaining 3DNS Controllers in the network, but be certain to choose **yes** when asked if you want NameSurfer to run as a master.

◆ **Note**

In this configuration you can perform zone edits on your secondary DNS server, and these changes will be broadcast to the primary DNS server. Normal BIND zone transfers will occur. If your primary DNS server becomes unavailable, your zone edits will not be broadcast.



D

DNS Resource Records



Overview

A resource record (RR) consists of a name, a type, and data that is specific to the type. These resource records, in a hierarchical structure, make up the DNS.

The standard resource record format, specified in RFC 1035, is as follows:

```
{name}    {ttl}    addr-class    record type    record-specific data
```

The fields are defined as follows:

❖ **name**

The first field, **name**, is the name of the domain record and it must always start in column 1. For all resource records that are not the first in a file, the name may be left blank. When the name field is left blank, the record takes the previous resource record.

❖ **ttl**

The second field, **ttl** (time to live), is optional. This field specifies how long this data will be stored in the database. If this field is left blank, the default time to live value is specified in the Start Of Authority resource record (described later in this chapter).

❖ **address class**

The third field is the address class. Currently, only one class is supported: **IN**, for internet addresses and other internet information. Limited support is included for the **HS** class, which is for MIT/Athena "Hesiod" information.

❖ **record type**

The fourth field, record type, defines the type of this resource record, such as "A."

❖ **other fields**

Additional fields may be present in a resource record, depending on its type.

Although case is preserved in names and data fields when loaded into the name server, comparisons and lookups in the name server database are case insensitive.

Types of resource records

There are many types of resource records currently in use. This section provides an overview of the most common resource record types, and lists other types of resource records.

Common types

There are six standard types of resource records:

Type	Description
A (Address)	Converts host names to IP addresses.
CNAME (Canonical Name)	Defines a host alias.
MX (Mail Exchange)	Identifies where to send mail for a given domain name.
NS (Name Server)	Identifies a domain's name servers.
PTR (Pointer)	Converts IP addresses to host names.
SOA (Start of Authority)	Marks the beginning of a zone's data, defines default parameters for a zone.

Table D.1 *Standard resource records*

A (Address)

The Address record, or **A** name record, lists the address for a given machine. The name field is the machine name, and the address is the network address. There should be one **A** name record for each address of the machine.

The following is an example of an **A** name record:

```
{name}      {ttl}  addr-class  A    address
ucbarpa      IN      A      128.32.0.4
              IN      A      10.0.0.78
```

CNAME (Canonical Name)

The Canonical Name resource record, CNAME, specifies an alias or nickname for the official, or canonical, host name. This record must be the only one associated with the alias name. It is usually easier to supply one **A** record for a given address and use CNAME records to define alias host names for that address.

The following is an example of a CNAME resource record:

```
alias    {ttl}    addr-class    CNAME    Canonical name
ucbmonet          IN              CNAME    monet
```

MX (Mail Exchange)

The Mail Exchange resource record, MX, defines the mail system(s) for a given domain.

The following is an example of an MX resource record:

```
name {ttl}  addr-class  MX  pref value  mail exchange
Munnari.OZ.AU.  IN      MX    0    Seismo.CSS.GOV.
*.IL.           IN      MX    0    RELAY.CS.NET.
```

NS (Name Server)

The Name Server resource record, NS, defines the name server(s) for a given domain, creating a delegation point and a subzone. The first name field specifies the zone that is serviced by the name server that is specified by the second name. Every zone needs at least two name servers.

The following is an example of an NS resource record:

```
{name}    {ttl}    addr-class    NS    Name servers name
          IN              NS    ucbarpa.Berkeley.Edu.
```

PTR (Pointer)

A Name Pointer record, PTR, associates a host name with a given IP address. These records are used for reverse name lookups.

The following example of a PTR record is used in setting up reverse pointers for the special IN-ADDR.ARPA domain:

name	{ttl}	addr-class	PTR	real name
7.0		IN	PTR	monet.Berkeley.Edu.

SOA (Start of Authority)

The Start of Authority, SOA, record starts every zone file. There must be exactly one SOA record per zone.

The following is an example of an SOA resource record:

name	{ttl}	addr-class	SOA	Origin	Person in charge
@		IN	SOA	ucbvax.Berkeley.Edu.	
kjd.ucbvax.Berkeley.Edu. (					
				1995122103	; Serial
				10800	; Refresh
				1800	; Retry
				3600000	; Expire
				259200	; Minimum
)					

The record-specific fields are defined as follows:

❖ **Person in charge**

The email address for the person responsible for the name server, with "@" changed to a ".".

❖ **Serial number**

The version number of this data file; it must be a positive integer. This number must be increased whenever a change is made to the data.

❖ **Refresh**

The time interval, in seconds, between calls that the secondary name servers make to the primary name server to see if an update is necessary.

❖ **Retry**

The time interval, in seconds, that a secondary server waits before retrying a failed zone transfer.

❖ **Expire**

The maximum number of seconds that a secondary name server can use the data before it expires for lack of receiving a refresh.

❖ **Minimum**

The default number of seconds to be used for the time to live (TTL) field on resource records which do not specify a TTL in the zone file. It is also an enforced minimum on TTL if it is specified on a resource record in the zone.

Other types

The following is a list of less common resource record types:

Type	Description
AAAA	IPv6 address
AFSDB	AFS database location
GPOS	Geographical position
HINFO	Host information
ISDN	Integrated services digital network address
KEY	Public key
KX	Key exchanger
LOC	Location information
MB	Mailbox domain name
MINFO	Mailbox or mail list information
NULL	A null RR
NSAP	Network service access point address
NSAP-PTR	(Obsolete)
NXT	Next domain
PX	Pointer to X.400/RFC822 information
RP	Responsible person
RT	Route through
SIG	Cryptographic signature
SRV	Server selection
TXT	Text strings
WKS	Well-known service description
X25	X25

Table D.2 Other types of resource records



Glossary



3DNS Maintenance menu

A command line utility that you can use to manually configure the 3DNS Controller.

3DNS web server

A standard web server that runs on the 3DNS Controller and hosts the Configuration utility, and also provides access to useful downloads.

A record

The answer that a 3DNS Controller returns to a local DNS server in response to a name resolution request. The **A** record contains a variety of information, including one or more IP addresses that resolve to the requested domain name.

active unit

In a redundant system, the controller that currently load balances connections. If the active unit in the redundant system fails, the standby unit assumes control and begins to load balance connections.

alternate method

Specifies the load balancing mode to use if the primary method fails.

big3d agent

A monitoring agent that collects metrics information about server performance and network paths between a data center and a specific local DNS server. The 3DNS Controller uses the information collected by the **big3d agent** for dynamic load balancing.

BIND (Berkley Internet Name Domain)

The most common implementation of DNS, which provides a system for matching domain names to IP addresses.

chain

A series of filtering criteria used to restrict access to an IP address. The order of the criteria in the series determines how the filter is applied, from the general criteria first, to the more detailed criteria at the end of the chain.

Configuration utility

The browser-based application that you use to configure the 3DNS Controller.

completion rate

The percentage of packets that a server successfully processes during a given conversation.

data center

A physical location that houses one or more 3DNS Controllers, BIG/ip Controllers, or host machines.

discovery factory

A tool managed by the **big3d** agent that checks for alternate ports to ping when trying to collect path data for a local DNS.

dynamic load balancing modes

Dynamic load balancing modes base connection distribution on live data, such as current server performance and current connection load.

dynamic ratio

An option for the Quality of Service load balancing mode. The dynamic ratio feature uses qos scores as ratios for virtual servers and distributes connections according to the ratio weight for each virtual server.

F-Secure SSH

An encryption utility that allows secure shell connections to a remote system.

fail-over

The process in which a standby unit in a redundant system takes over due to a software or hardware failure detected on the active unit.

fail-over cable

The cable that directly connects the two controller units in a redundant system.

FDDI (Fiber Distributed Data Interface)

A multi-mode protocol for transmitting data on optical-fiber cables up to 100 Mbps.

First-Time Boot utility

A utility that walks you through the initial system configuration process. The First-Time Boot utility runs automatically when you turn on a controller for the first time.

fallback method

Specifies the last load balancing mode that the 3DNS Controller tries to use if both the primary and the alternate methods fail.

hardware-based fail-over

A redundant system in which the two units are connected directly by a cable.

hops

One point-to-point transmission in a network path between a host and a client server. A network path that included a stop at a network router would have two hops: the first from the client to the router, and the second from the router to the host server.

hops factory

A type of factory run by the **big3d** agent that collects hops data for network paths.

host

A network server which manages one or more virtual servers that the 3DNS Controller uses for load balancing.

ICMP (Internet Control Message Protocol)

An Internet communications protocol used to determine information about routes to destination addresses, such as virtual servers managed by BIG/ip Controllers and 3DNS Controllers.

IP filter

A filter that specifically allows or denies traffic passing through an interface on the 3DNS Controller, based on either the source IP address, the destination IP address, or both.

iQuery

A UDP-based protocol used to exchange information between **big3d** agents and 3DNS Controllers. The iQuery protocol is officially registered for port 4353.

LDNS probe state

The status of an local DNS server with respect to metrics collection.

LDNS Round Robin

A standard DNS feature that allows a DNS server, or a 3DNS Controller, to return multiple IP address in the **A** record. The local DNS, or even the browser, can cache the IP addresses and use them for future name resolutions.

local DNS (LDNS)

A local DNS server typically found at a client's Internet service provider. The 3DNS Controller calculates path information for the path between the local DNS and the virtual servers that the 3DNS Controller is load balancing, and also uses the local DNS IP address for topology-based load balancing and access control.

master DNS

A DNS server that is considered authoritative for one or more DNS zones. See also *zones*.

metrics

Performance data, including server performance and network path integrity, collected by **big3d** agents and used by the 3DNS Controller for dynamic load balancing.

named

The name server daemon, which manages domain name server software.

NameSurfer

The third-party application that automatically manages DNS zone files on 3DNS Controllers.

network-based redundant system

A redundant system in which two units communicate over a network connection, rather than a hard-wired connection.

path

The network route between two specific IP addresses. For dynamic load balancing, the 3DNS Controller uses information about the path between a client LDNS and a specific virtual server that it is load balancing.

packet rate

The number of packets per second going in or coming out of a given server.

pool

A group of virtual servers to which the 3DNS Controller distributes connections when load balancing a specific domain.

port

A number that is associated with a specific service supported by a host.

preferred method

Specifies the primary load balancing mode used to load balance a wide IP.

principal 3DNS Controller

A 3DNS Controller that initiates metrics collection by the **big3d** agents. Note that a sync group can have only one principal.

probing factory

A tool managed by the **big3d** agent that queries virtual servers to determine whether they are **up** or **down**, and also to determine path metrics such as round trip time and hops.

production rule

A tool that can change system behavior under specific operating conditions. For example, a production rule can switch load balancing modes or can reroute network traffic to a specific set of servers based on triggers such as time of day or current network traffic load.

qos coefficient

The ratio weight for a specific factor used in a **qos** equation, such as hops, round trip time, packet rate, completion rate, or topology score. Each factor has a default coefficient of 1, but you can assign user-defined coefficients to put more weight on a specific factor, and less on other factors.

qos equation

A calculation based on various path statistics used for dynamic load balancing including hops, round trip time, packet rate, completion rate, and topology score.

qos score

The result of a **qos** calculation. When using dynamic load balancing modes, the 3DSN Controller uses the **qos** score to determine which virtual server has the best performance and should receive new connections.

ratio

A parameter that assigns a weight to a virtual server for load balancing purposes.

Ratio mode

The Ratio load balancing mode distributes connections across an array of virtual servers in proportion to the ratio weights assigned to each individual virtual server.

receiver 3DNS Controller

A 3DNS Controller that receives broadcasted metrics data from **big3d** agents, but does not initiate metrics collection (see ***principal 3DNS Controller***).

redundant system

A pair of controllers that are configured for fail-over. In a redundant system, there are two controller units, one running as the active unit and one running as the standby unit. If the active unit fails, the standby unit takes over and manages connection requests.

remote administrative IP address

An IP address from which a controller allows shell connections, such as Telnet or SSH.

rsetup

The script that configures the remote login tools, **rsh** and **rcp**, on a controller.

slave DNS server

A DNS server that updates its records by querying a master DNS server.

shared IP alias

The IP address that represents a redundant system.

SNMP (Simple Network Management Protocol)

A standard Internet standard protocol developed to manage nodes on an IP network.

SNMP agent

The agent that controls SNMP communications on a server.

SNMP factory

A type of factory run by **big3d** agents that uses the SNMP protocol to collect metrics data for host servers.

SNMP MIB (Management Information Base)

A text file in standard SNMP format that defines the individual objects you can manage with common SNMP tools such as the see/IT Network Manager or HP OpenView.

sod (switch over daemon)

A daemon that controls the fail-over process in a redundant system.

standby unit

A controller in a redundant system that is always prepared to become the active unit if the active unit fails.

static load balancing mode

A static load balancing mode bases connection distribution on a pre-defined list of criteria; it does not take current server performance or current connection load into account.

sync group

A group of 3DNS Controllers that share configuration information. Each sync group has one principal 3DNS Controller, and can also have one or more receiver 3DNS Controllers.

time tolerance value

The number of seconds that one 3DNS Controller's clock is allowed to differ in comparison to another 3DNS Controller's clock, without the two clocks being considered out of sync.

topology record

A record that specifies a score for a specific pair of a local DNS IP address and a virtual server IP address or range of IP addresses. Topology records are used for topology-based access control, and topology load balancing modes.

topology score

A rank for a specific pair of a local DNS server and a virtual server IP address, or range of IP addresses.

topology-based access control

A method of specifically allowing or preventing clients from connecting to particular virtual servers based on the IP address of the requesting client's local DNS and the IP address of the virtual server.

traceroute

The utility that the hops factory uses to calculate total number of network hops between an LDNS and a specific data center.

TTL (time-to-live)

The number of seconds for which a specific DNS record or metric is considered to be valid. When a TTL expires, the server usually must refresh the information before using it.

virtual server

A specific combination of a virtual address and virtual port, associated with a content site that is managed by a BIG/ip Controller or other type of host server.

watchdog timer card

A hardware device that monitors the 3DNS Controller for hardware failure.

well known services (WKS)

A type of resource record that describes the services usually provided by a particular protocol on a particular port.

wide IP

A mapping of a fully-qualified domain name to a set of virtual servers that host the domain content, such as a web site or an e-commerce site.

wide IP key

An IP address associated with a wide IP. The wide IP key is the same address that standard DNS associates with the domain name the wide IP load balances.

zone

A subset of DNS records for one or more domains.

zone file

A set of DNS records that contain domains with one or many domain names, designated mail servers, a list of other name servers that can answer resolution requests, and a set of zone attributes called SOA (Start Of Authority).



Index



/etc/bigip.conf 3-19
 /etc/daily B-9
 /etc/ethers 3-19
 /etc/hosts.allow 3-9, 7-27
 /etc/hosts.equiv 3-9
 /etc/netstart 3-19
 /etc/snmpd.conf 7-28
 /etc/sshd_config 3-19
 /var/f5/httpd/basicauth/users 3-17
 /var/f5/httpd/conf/httpd.conf 3-19
 /var/log/3dns B-9
 /var/log/messages 5-9

100baseTX 3-14
 100baseTX,FDX 3-14
 10baseT 3-14
 10baseT,FDX 3-14
 3DNS Controller
     welcome 1-1
 3DNS Maintenance menu 1-4
     Install and Start big3d 2-11
 3DNS web server 1-4, 3-16
     changing configuration 3-17
     changing the password 7-33
     password file 7-34
 3dns_admin_start B-31
 3dns_auth B-31
 3dns_dump B-33
 3dns_sync_metrics B-33
 3dns_web_config B-33
 3dns_web_passwd B-33
 3dnsmaint 7-2, B-34
 3dparse B-1
 3dparse.out B-2
 3dprint B-34
 3ndc B-4, B-35

A

A records 1-11
 A resource records D-2

active unit 2-5
 administering 3DNS Controllers 1
 allocating subdomains C-13
 authentication certificate
     entering information 3-16
 availability
     virtual servers 7-54

B

BIG/config 1-3
 BIG/ip Controller
     connection management 1-15
 BIG/ip Controllers
     defining 4-9
     enabling remote login tools 3-20
     statistics 7-46
 big3d agent 1-5
     sample configuration 1-10
 big3d agents 2-10, 4-3
     configuring 2-9
     installing B-35
     installing on BIG/ip Controllers 2-9, 2-11
 big3d_check B-35
 big3d_install B-35
 big3d_restart B-36
 big3d_version B-36
 BIND
     converting to NameSurfer 2-6
 BIND 8 C-1
     comments C-8
     configuration files C-1
     converting older files C-10
 BIND 8 statements C-1
     acl C-2
     key C-3
     logging C-4
     options C-5
     server C-6
     zone C-7
 broadcast addresses
     interfaces 3-12
 broadcasting

- big3d agents 1-10, 1-18
- sync groups 1-17
- to redundant systems 2-5
- browsers, supported versions 1-4
- BSD IP packet filtering 2-37

C

- checkpoints B-38
- CNAME resource records D-3
- command line access
  - configuring 3-15
- comments
  - 3DNS Controller A-48
  - 3DNS Controller syntax A-49
  - BIND 8 C-8
  - BIND 8 syntax C-8
- completion rate
  - load balancing mode 2-32
- Completion Rate mode 1-9
- config-httpd 3-17
- configuration files 3-19
  - 3DNS A-1
  - BIND 8 C-1
  - example (3DNS Controller) A-50, A-64
  - requirements A-1
  - statements A-4
  - syntax, comments A-49
  - syntax, datacenter statement A-25
  - syntax, globals statement A-7
  - syntax, server statement (3DNS type) A-29
  - syntax, server statement (BIG/ip type) A-31
  - syntax, server statement (host type) A-33
  - syntax, sync_group statement A-27
  - syntax, topology statement A-47
  - syntax, wideip statement A-40
- configuring
  - 3DNS Controllers 4-6
  - BIG/ip Controllers 4-9
  - data centers 4-2
  - global variables 4-29
  - hosts 4-14

- IP filters 4-31
- servers 4-5
- sync groups 4-27
- syslog for 3DNS messages B-7
- wide IPs 5-2
- creating empty 3DNS Controller files B-8
- cur_values A-78

D

- data centers 1-17
  - configuring 4-2
  - planning 2-7
  - statistics 7-42
- datacenter statement A-25
- default route 3-9
  - configuring 3-11
- delegating subdomains C-13
- destination IP address 2-37
- DNS
  - master servers 1-11
  - root servers 1-12
- DNS zone file management
  - setting up 3-18
- DNS zone files
  - installation issues 2-6
  - planning 2-38
- domain names, maximum supported 1-3
- dynamic load balancing 1-8
- dynamic ratio 1-19
- Dynamic Ratio attribute 6-6

E

- e-commerce, configuring wide IPs
  - ftp, configuring wide IPs 2-35
- edit_lock B-37
- edit_wideip B-37
- editing wideip.conf A-1
- encryption
  - enabling 4-30
  - using f5makekey B-73
- Ethernet 3-2
- examples

- 3DNS Controller configuration file A-50, A-64
- syntax for datacenter statement A-26
- syntax for globals statement A-10
- syntax for server statement (3DNS type) A-30
- syntax for server statement (BIG/ip type) A-32
- syntax for server statement (host type) A-34
- syntax for sync_group statement A-27
- syntax for wideip statement A-41
- Topology mode 6-23
- exp0 3-12

F

- F5 Configuration utility 1-3, 1-16, 2-8
- F5 technical support
  - web server access 3-16
- f5makekey B-37
- fail-over 3-16
  - hardware-based 1-6
  - network-based 1-6
  - triggers 2-5
- fail-over cable 3-1, 3-7
- fan filters 3-1
- FDDI 3-14
- FDDI/CDDI 3-2
- firewalls
  - configuring for 2-22
- First-Time Boot utility 1-3, 2-3, 3-8
  - confirming settings 3-18
  - editing settings 3-18
  - saved files 3-19
  - saving settings 3-19

G

- Gigabit Ethernet 3-2, 3-14
- Global Availability mode 1-7, 6-8
- global variables
  - configuring 4-29
  - related to load balancing 5-9

- globals statement A-6

H

- hardware
  - components 3-1
  - rack mounting requirements 3-4
  - redundant systems 1-16
  - setup 2-1
  - temperature control 3-5
  - voltage selector 3-5
- hardware fail-over 1-16
- hardware setup
  - planning 2-4
- hardware-based fail-over 1-6
  - installation issues 2-5
- hops coefficient, qos equation 1-18
- Hops load balancing mode 1-18
- Hops mode 1-8
- host name, of controller 3-11
- host names
  - in First-Time Boot utility 3-9
- hosts
  - configuring 4-14
  - probing 2-14, 4-14
  - statistics 7-50
  - supported SNMP agents 2-15
- httpd.conf 3-17

I

- include files A-2
- include statements 1-19
- Install and Start big3d command 2-11
- install_key B-37
- installation
  - hardware setup 2-1
  - network setup 2-1
  - phases, overview 2-1
  - rack mounting 3-4
- interfaces
  - using multiple 2-6
- international configurations 3-9
- Internet Explorer 1-4

- IP filters 1-3, 1-6
  - configuring 4-31
  - destination IP addresses 4-32
  - source IP addresses 4-32
- IP packet filtering 2-36, 2-37
- iQuery 1-2
  - configuring for firewalls 2-22

K

- key
  - generating for encryption B-37
- keys, front panel 3-1

L

- LDNS Round Robin 1-19, 2-36
- least connections
  - load balancing mode 2-32
- Least Connections mode 1-9
- load balancing 1
  - Completion Rate mode 1-9
  - configuration planning 2-2
  - dynamic modes 1-8
  - dynamic ratio attribute 1-19
  - Dynamic Ratio option 6-6
  - Global Availability mode 1-7, 6-8
  - Hops mode 1-8, 1-18
  - LDNS round robin attribute 1-19
  - Least Connections mode 1-9
  - Packet Rate mode 1-9
  - pools 1-13
  - Quality of Service mode 1-8, 6-1
  - Random mode 1-7
  - Ratio mode 1-7
  - related global variables 5-9
  - Round Robin mode 1-7
  - Round Trip Times mode 1-8
  - static modes 1-7
  - Topology mode 1-8, 6-22
- load balancing modes
  - completion rate 2-32
  - least connections 2-32
  - null 2-30

- packet rate 2-33
- random 2-29
- ratio 2-29
- return to DNS 2-31
- round robin 2-28
- round trip times 2-34

- local DNS
  - statistics 7-55
- log rotation B-9
- log2mail B-13
- logging
  - configuring B-7

M

- maintenance menu 7-1
- master DNS
  - using NameSurfer 3-18
- media types 3-14
- metrics
  - collecting from hosts 2-14
- MS resource records C-15
- MX resource records D-3

N

- name resolution 1-11, 1-12
- named 2-5, B-5
- NameSurfer 1-4, 1-16
  - configuring 2-6
  - converting from BIND 2-6
  - setting up 3-18
  - transferring from BIND C-21
- ndc
  - as an alias B-4
- ndc restart 2-5
- ndc stop 2-5
- netmasks
  - interfaces 3-12
- Netscape 1-4
- network fail-over 1-17
- network interface cards 1-17, 2-4, 3-2
  - configuring 3-11
  - configuring multiple 2-6

- fail-over triggers 2-5
- media types 3-14
- network setup 2-1
- network statistics 7-35
- network-based fail-over 1-6
  - installation issues 2-5
- new features 1-16
- NS resource records D-3
- null
  - load balancing mode 2-30

O

- online help 1-20

P

- packet rate
  - load balancing mode 2-33
- Packet Rate mode 1-9
- password authentication B-31
- passwords
  - 3DNS web server 7-33
  - in First-Time Boot utility 3-9
  - root user, defining 3-10
  - web server 3-16
- path probing
  - SNMP-based 1-19
- paths
  - statistics 7-54
- PC/AT-to-PS/2 keyboard adapter 3-1, 3-6
- PDF version, Administrator Guide 1-20
- peer IP address 3-13
- periodic task intervals A-13
- pools 1-13, 5-3, A-42
- primary Ethernet interface 3-6
  - configuring 3-11
- principal 3-18
- principal 3DNS Controller 1-11
- probers
  - statistics 7-48
- probing
  - hosts 2-14, 4-14
- production rules 1-18, 2-36, 2-38

- PTR resource records D-4

Q

- Quality of Service mode 1-8, 6-1
  - balancing coefficients 6-2

R

- rack mounting 3-4
- rack-mounting screws 3-1
- random
  - load balancing mode 2-29
- Random mode 1-7
- ratio
  - load balancing mode 2-29
- Ratio mode 1-7
- rcp
  - enabling on BIG/ip Controllers 3-20
- receiver 3-18
- redundant system
  - configuration changes 2-6
- redundant systems 1-6, 1-16
  - active unit 2-5
  - as sync group members 2-5
  - configuring shared IP alias 3-12
  - fail-over triggers 2-5
  - installation issues 2-4
  - standby unit 2-5
- release notes 1-19
- remote administration 2-3, 2-8, 3-9
  - configuring 3-15
  - downloading SSH client 3-20
- requirements
  - configuration files A-1
- resource records 1
  - A D-2
  - CNAME D-3
  - MX D-3
  - NS D-3
  - PTR D-4
  - SOA D-4
- return to DNS
  - load balancing mode 2-31

- reverse domains C-15
- root password
  - defining 3-10
- root user
  - password 3-9
- round robin
  - load balancing mode 2-28
- Round Robin mode 1-7
- round trip times
  - load balancing mode 2-34
- Round Trip Times mode 1-8
- RSA authentication
  - generating B-31
- rsetup script 3-20
- rsh 3-9
  - configuring remote administration 3-15
  - enabling on BIG/ip Controllers 3-20

S

- sample configuration 1-9
  - big3d agent communications 1-10
  - network layout 1-10
- scripts B-31
  - 3dns_admin_start B-31
  - 3dns_auth B-31
  - 3dns_dump B-33
  - 3dns_sync_metrics B-33
  - 3dns_web_config B-33
  - 3dns_web_passwd B-33
  - 3dnsmaint B-34
  - 3dprint B-34
  - 3ndc B-35
  - big3d_check B-35
  - big3d_install B-35
  - big3d_restart B-36
  - big3d_version B-36
  - edit_lock B-37
  - edit_wideip B-37
  - install_key B-37
  - syncd_checkpoint B-38
  - syncd_rollback B-39
  - syncd_start B-40
  - syncd_stop B-41
- secondary Ethernet interface 3-6
  - configuring 3-11, 3-14
- security
  - changing passwords 7-33
- sending mail from syslog B-13
- Sendmail
  - configuring 4-34
- serial terminal 2-3, 3-2, 3-6
- server statement A-28
- servers
  - configuring 4-5
  - defining 2-7, 2-8
- shared IP alias 2-4, 3-12
  - configuring broadcast 3-12
  - configuring netmask 3-12
- SMTP 1-2
- SNMP 1-2
  - client access 7-28, 7-30
  - host probing 2-14
  - in F5 Configuration utility 7-30
  - MIB 7-26, 7-30
  - path probing for hosts 1-19
  - probing hosts 4-14
  - system contacts 7-31
  - trap configuration 7-29, 7-31
- SNMP agents
  - on hosts 4-20
  - supported for host probing 2-15
- SNMP factory 2-14
- SNMP MIB 1-4, 1-16
- SOA resource records D-4
- sod-named B-3
- source IP address 2-37
- SSH 1-2, 3-9
  - configuring remote administration 3-15
- SSH client
  - downloading 3-20
  - manual 3-1
  - UNIX 3-24
  - Windows 95 and Windows NT 3-23
- SSL 1-2
- standby unit 2-5
- statements
  - 3DNS Controller A-4

- BIND 8 C-1
  - datacenter A-25
  - globals A-6
  - server A-28
  - sync_group A-27
  - topology A-46
  - wideip A-39
  - static load balancing 1-7
  - statistics
    - 3DNS Controllers 7-42, 7-46
    - BIG/ip Controllers 7-46
    - data centers 7-42
    - hosts 7-50
    - local DNS 7-55
    - path 7-54
    - probers 7-48
    - summary 7-35
    - sync groups 7-43
    - virtual servers 7-52
    - wide IPs 7-44
  - sub-domains 2-38
  - subdomains
    - allocating C-13
    - delegating C-13
  - summary
    - network statistics 7-35
  - sync groups 1-17
    - broadcasting configurations 2-7
    - configuring 4-27
    - defining 3-17
    - planning 2-8
    - redundant systems 2-5
    - sample configuration 1-11
    - statistics 7-43
  - sync_group statement A-27
  - syncd_checkpoint B-38
  - syncd_rollback B-39
  - syncd_start B-40
  - syncd_stop B-41
  - syntax
    - acl statement (BIND 8) C-3
    - comments (3DNS Controller) A-49
    - comments (BIND 8) C-8
    - datacenter statement (3DNS Controller) A-25
    - globals statement (3DNS Controller) A-7
    - key statement (BIND 8) C-3
    - logging statement (BIND 8) C-4
    - options statement (BIND 8) C-5
    - rules A-5
    - server statement (3DNS type) A-29
    - server statement (BIG/ip type) A-31
    - server statement (BIND 8) C-6
    - server statement (host type) A-33
    - sync_group statement (3DNS Controller) A-27
    - topology statement (3DNS Controller) A-47
    - wideip statement (3DNS Controller) A-40
    - zone statement (BIND 8) C-7
  - syslog 7-30
    - configuring for 3DNS messages B-7
  - syslog.conf B-9
  - syslogd B-12
- ## T
- Technical Support web site 1-20
  - temperature control 3-5
  - time tolerance value 4-29
  - time zone, configuring 3-11
  - Topology access control 6-13
  - Topology mode 1-8, 6-22
  - topology statement A-46
  - Topology-based access control 2-36
  - trap configuration 7-30
  - troubleshooting
    - configuration problems 5-8
- ## U
- user ID
    - web server 3-16
  - utilities
    - 3dparse B-1

- First-Time Boot 1-3, 3-8
- log2mail B-13
- syslogd B-12

V

- verifying wideip.conf syntax B-1
- VGA monitor 3-6
- virtual servers
  - criteria for availability 7-54
  - defining 2-8
  - statistics 7-52
- voltage selector 3-5

W

- web server 3-16
  - allowing F5 access 3-16
  - downloads 3-21
  - password 3-9
- web server, see 3DNS web server 1-4
- wide IP sub-domains 2-38, 2-41
- wide IPs
  - configuring 5-2
  - statistics 7-44
- wideip statement A-39
- wideip.conf A-1
  - example A-50, A-64
  - include statements 1-19
  - requirements A-1
  - verifying syntax B-1

Z

- zone files
  - installation issues 2-6
  - transferring to 3DNS 2-39
  - transferring to NameSurfer C-21
- zone minimums
  - overriding C-20
  - specifying C-20