f5. NETWORKS

3 DNS.

Administrator Guide

Administrator Guide for the 3DNS® Controller

version 1.0.6

Service and Support Information

Product Version

This manual applies to version 1.0.6 of the 3DNS® Controller. To obtain technical support for these products, or to request product sales or customer service information, refer to the contact information provided below.

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

Information furnished by F5 Networks, Inc. (F5) is believed to be accurate and reliable. However, no responsibility is assumed by F5 for its use, nor any infringements 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 describes herein. F5 reserves the right to change specifications at any time without notice.

Trademarks

F5, 3DNS, and BIG/ip are registered 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 is shipped with cryptographic software. Therefore, under the Export Administration Act, the United States government may consider it a criminal offense to export this 3DNS Controller from the United States.

FCC Compliance

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

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 John Kohl.

This product includes software developed by Paul Kranenburg.

This product includes software developed by Terrence R. Lambert.

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

This product includes software developed by Philip A. Nelson.

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 the 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 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 Stage 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.

This product includes software developed by Dean Huxley.

This product includes software developed by Herb Peyerl.

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

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

This product includes software developed by Thomas Boutell (boutell@boutell.com).

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."

The material included Appendix E, *BIND 8 Configuration Information*, 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: (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 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, M-F, 6 am - 6 pm 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 (e.g. 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 (e.g. 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 US government regulations relating to the export of technical data and equipment and products produced therefrom, which are applicable to Licensee. In countries other than the US, Licensee agrees to comply with the local regulations regarding 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 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 (a) has been altered, except by F5, (b) has not been installed, operated, repaired, or maintained in accordance with F5's instructions, (c) has been subjected to abnormal physical or electrical stress, misuse, negligence or accident or (d) has been operated outside of the environmental specifications for the product. F5's limited software warranty does not apply to software corrections or upgrades.

6. 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, 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

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Chapter 8

Appendix A

Appendix B

Appendix C

Appendix D

Appendix E

Appendix F

1

Introduction to 3DNS Controller

- Welcome to the 3DNS Controller
- Features
- What’s new in this version
- Conventions used in this manual

Welcome to the 3DNS Controller

The 3DNS® Controller is a wide area load distribution solution. It works in tandem with BIG/ip® Server Array Controllers, other server array controllers, and single network servers to intelligently allocate Internet and intranet service requests across a geographically distributed array of network servers. The 3DNS Controller provides intelligent name resolution and adds load balancing intelligence to the latest BIND technology. Using the 3DNS Controller, you can provide clients with optimal performance, the most current data, safe data access, high availability, and protection from failed systems.

Figure 1.1 shows how 3DNS Controllers fit into a global network.

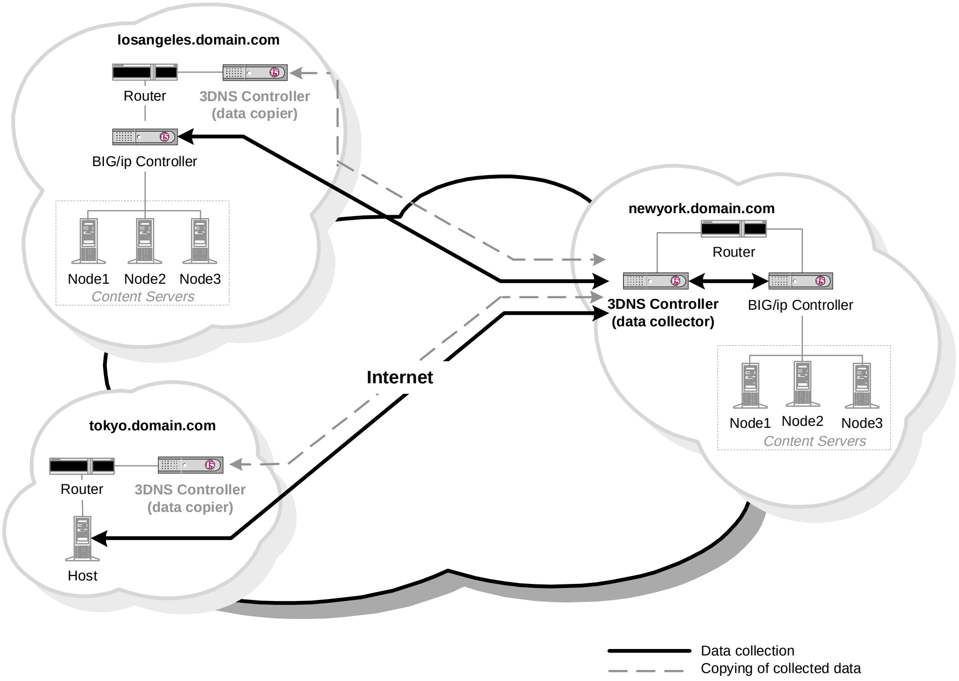


Figure 1.1 *Using 3DNS Controllers on a global network*

The network in Figure 1.1 uses the following configuration:

- The client machine uses an Internet Service Provider (ISP) located in Chicago to connect to the local DNS, which in turn connects to the primary DNS.
- The primary DNS can be outside of the customer network, as shown here, or you can configure a 3DNS Controller to be the primary DNS within the customer network. In this example, name resolution requests for specified domains are delegated from the primary DNS to the 3DNS Controller that is the data collector. For step-by-step descriptions of the name resolution process, see pages 2-3 through 2-8.
- 3DNS Controllers are installed in New York, Los Angeles, and Tokyo.
- The 3DNS Controller in New York is the data collector. As data collector, it gathers performance data by querying the BIG/ip Controllers in New York and Los Angeles, and the host machine in Tokyo.
- The 3DNS Controllers in Los Angeles and Tokyo are data copiers. As data copiers, they copy performance data from the data collector (the 3DNS controller in New York) and store the collected data in their caches, in case the data collector fails.
- The data collector resolves name resolution requests using the performance data and a load balancing algorithm. For details on the available load balancing modes, see Chapter 5, *Load Balancing*.

Note

Some countries do not allow data encryption. An international version of the 3DNS Controller is available for these situations. For more information, see Working with international versions, on page 2-15.

Features

With 3DNS Controllers properly implemented on a geographically dispersed network, the network becomes more efficient, reliable, and scalable.

Efficiency

3DNS Controllers increase efficiency of a network in the following ways:

- **Performance**
  Maximizes access performance by providing highly available, transparent, IP services.
- **Intelligent routing**
  Provides intelligent traffic routing with advanced load balancing algorithms: Completion Rate, Global Availability, Least Connections, Packet Rate, Quality of Service (QOS), Random, Ratio (also known as Weighted or Administrative Cost), Round Robin (RR), Round Trip Time (RTT), and Topology.
- **Ease of integration**
  Integrates seamlessly with BIG/ip Controllers. Also integrates with other array controllers, as well as individual network servers.
- **Collecting information**
  The 3DNS Controller collects information, allowing the 3DNS Controller to answer subsequent requests from a local DNS more intelligently. Answers are returned immediately. The 3DNS Controller does not collect information as a result of or during the name resolution process. Instead, the 3DNS Controller collects information at pre-configured intervals. With the 3DNS Controller, you can specify how long data is saved in the cache. For example, by specifying low time to live (TTL) values, you ensure that client requests are satisfied with the most current data, rather than with existing data from the cache.

Reliability

It is important to ensure that clients have access to the services they need at all times. The following features ensure the reliability of a network:

- **Adherence to standards**
  The 3DNS Controller is based on industry-standard DNS.
- **Transparent distribution**
  The 3DNS Controller allows transparent distribution of all IP services.
- **Encryption**
  3DNS Controllers distributed only in the US provide support for Blowfish CBC encryption, which keeps iQuery protocol transactions secure. The iQuery protocol is the protocol used to communicate and exchange information between BIG/ip Controllers and 3DNS Controllers. Note that 3DNS Controllers distributed outside the US do not support encryption.

Scalability

3DNS Controllers provide the flexibility to effectively manage changing network demands. With 3DNS Controllers in place, your network becomes more scalable by:

- Allowing servers and BIG/ip Controller clusters to be transparently added or removed.
- Supporting an unlimited number of distributed content servers and array controllers.
- Leveraging BIG/ip Controller’s ability to handle all servers in a local array as a single IP address.

What’s new in this version

The following features are new in version 1.0.6 of the 3DNS Controller.

New load balancing options

The 3DNS Controller now supports three hierarchical load balancing methods. For each pool in a `wideip` statement, you can specify a preferred method, an alternate method, and a fallback method. See *The wide IP statement*, on page 7-21.

Topology-based access control

3DNS Controller can now control access to specific data centers, based on the IP address of the requesting local DNS. See *Topology-based access control*, on page 5-15.

New static load balancing mode: Topology

The new Topology load balancing mode distributes connections based on the proximity of a local DNS to a particular data center. See page 5-21. The topology mode can also be incorporated into the Quality of (QOS) load balancing mode.

New distribution method: e-commerce

Using the **port_list** parameter, you can configure a wide IP so that connections are not sent to a given address unless all listed services are available. This feature is especially useful for e-commerce transactions. See *E-commerce*, on page 5-22.

New versions of big3d

3DNS Controller includes a new big3d utility for all versions of BIG/ip Controller.

Enhancements to the 3DNS Web Administration tool

The 3DNS Web Administration tool now includes an Administration area where you can change the 3DNS Controller configuration and control statistics collection. The original statistics screens also contain new information in several areas. See Chapter 6, *Web Administration*.

3DNS Maintenance menu changes

The 3DNS Maintenance menu includes several new commands:

- Check versions of named, BIG/ip kernel and needed big3d
- Edit big3d matrix
- Dump and List named database
- Display mode of wideip.conf
- Use Static wideip.conf

- Use Dynamic wideip.conf

See *The 3DNS Maintenance menu*, on page 4-23.

iQuery enhancements

3DNS Controller has three new iQuery options:

- **New port**
  The iQuery protocol is officially registered with the IANA for port 4353, and you can run iQuery on either that port or on the original port 245.
- **Port selection**
  You can distribute return iQuery traffic across individual ephemeral ports, or you can use either port 245 or 4353 as a single port for return iQuery traffic.
- **Translation**
  You can now set iQuery to include translated IP addresses in iQuery packets (useful for configurations where iQuery communication between a BIG/ip Controller and a 3DNS Controller passes through a firewall). See *Configuring iQuery options*, on page 4-20.

Improved path probing

3DNS Controller now has advanced path probing schemes, which determine path attributes such as round trip time and packet completion rate. See *Understanding probing*, on page 2-21.

Storing dynamic and static copies of the wideip.conf file

You can now store your original *wideip.conf* file separately from a *wideip.conf* file that stores current path and local DNS information. See *Working with static and dynamic wideip.conf files*, on page C-2.

Increasing storage space for zone files

You now have the option of storing zone files in a */var/namedb* directory, which offers substantially more storage space than the */etc/namedb* directory. See *Storing zone files*, on page 3-7.

New First-Time Boot utility trigger

In previous versions of 3DNS Controller, the First-Time Boot utility ran at start up if the system did not detect the */etc/wideip.conf* file. However, in the current version, the First-Time Boot utility is triggered only if the */etc/netstart* file is not found. The */etc/wideip.conf* file is no longer used to trigger or prevent the First-Time Boot utility from running at start up. If you are upgrading from an earlier version, you must change the appropriate lines in the */etc/rc* file to take advantage of this change. See *Upgrading an earlier version*, on page 3-4.

Comments are allowed in bigips.txt and 3dns.txt files

You can now use shell style comments (also known as Perl style comments) in the *bigips.txt* and *3dns.txt* files. See *File location*, on page D-20.

Support for international 3DNS Controllers

3DNS Controller now supports versions for international distribution. See page 2-15.

New utility: watchdog-named

You can use the new *watchdog-named* utility to start and monitor the *named* process. See *watchdog-named, on page D-3*. It is important to note that when your 3DNS Controller is using *watchdog-named*, you cannot use *ndc* to stop, start, or restart *named*. Instead, you must use *3ndc*. See *3ndc*, on page D-5.

Conventions used in this manual

This section describes the typographic and terminology conventions used in this manual.

Typographic conventions

Understanding these conventions is especially useful in learning command syntax.

Parameters

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.

Typeface

The courier typeface is used to distinguish user input and computer output from explanatory text.

- **Computer prompts, computer output, and file excerpts**
  Computer prompts, computer output, and file excerpts are shown in Courier type, as in:

```
globals {
   default_alternate ratio
```

- **User input**
  Text you must type is shown in bold Courier type, as in:

```
big3d -version
```

Terminology conventions

The following terms, used in this manual, require some explanation:

Host machine

The term ***host machine*** refers to an individual network server or server array controller other than the BIG/ip Controller.

Data collector/data copier

You can configure a 3DNS Controller to be a data collector or a data copier:

- **Data collector**
  A ***data collector*** is a 3DNS Controller that collects metrics information. By default, all 3DNS Controllers on a global network are peers, meaning that they each collect metrics information. A 3DNS Controller is a data collector until you specifically designate it to be data copier using the globals sub-statement `primary_ip`. See *Defining data collectors and data copiers*, on page 4-18.
- **Data copier**
  A ***data copier*** is a 3DNS Controller that copies metrics from a data collector at intervals specified with the globals sub-statement `sync_db_interval`. Data copiers do not collect metrics themselves.

DNS

The ***Domain Name System (DNS)*** is a distributed database that maps IP addresses to host names. All DNS servers (DNS and 3DNS) resolve names.

The terms *primary* and *secondary* are used to differentiate between DNS systems that maintain authoritative zone information, and DNS systems that copy zone information from other DNS systems:

- **Primary DNS**
  A ***primary DNS*** is the authoritative source for zone information. All DNS servers can resolve names, but zone files are kept and configured only on primary DNS servers.
- **Secondary DNS**
  A ***secondary DNS*** is a DNS server that is instructed to get its database from a primary DNS on a zone-by-zone basis. The secondary DNS copies zone files from the primary DNS at startup, when a timer expires in the SOA record, or when a dynamic update occurs.

This manual assumes that you have general knowledge of DNS. For complete documentation of DNS, you can refer to O'Reilly & Associates’ book *DNS and BIND* (second or third edition). When you review DNS documentation that covers BIND 8, you will notice that BIND 8 now uses the terms *master* and *slave* instead of *primary* and *secondary.*

◆ **Note**

You can configure a 3DNS Controller so that it handles DNS name resolution and authoritative zone information, in addition to metrics collection. In this case, the 3DNS machine is the data collector as well as the primary DNS.

Virtual server

The term "VIP" has been replaced by ***virtual server***, and it is used to refer to a specific combination of a virtual IP address and a virtual port number managed by a BIG/ip Controller or other host machine. Throughout this manual, virtual servers managed by BIG/ip Controllers are represented by ***vsb***, and virtual servers managed by other host machines are represented by ***vsh***.

Node

The term ***node*** refers to a specific combination of a node address and a node port number, which is managed by the BIG/ip Controller. A BIG/ip Controller maps each virtual server to one or more nodes. In the 3DNS Web Administration tool, *Nodes Up* denotes the number of nodes that are currently available for a given virtual server. The 3DNS Controller monitors and collects data for nodes that are managed only by BIG/ip Controllers.

Local DNS

The term ***local DNS*** refers to a DNS server that makes name resolution requests on behalf of a client. From the 3DNS Controller’s perspective, the local DNS is the source of the name resolution request.

2

Preparing for Installation

- **General network considerations**
- **Planning the primary DNS**
- **Integrating 3DNS Controllers**
- **Working with international versions**
- **Understanding virtual servers**
- **The iQuery protocol**
- **Setting up the big3d utility**
- **Understanding probing**
- **Port and protocol usage**

General network considerations

Before you install a 3DNS Controller, you should do some careful planning for your network. The issues you need to consider vary, depending on your network environment:

- Decide where the primary DNS should be located. Should it remain on its own machine, inside or outside of your network, or do you want to migrate the existing primary DNS to a 3DNS machine? See the following section, *Planning the primary DNS*.
- Decide how to integrate 3DNS Controllers and where to locate data collectors and data copiers. Note that all 3DNS Controllers are data collectors until you specify otherwise. See *Integrating 3DNS Controllers*, on page 2-8.
- If you are preparing to install BIG/ip® Controllers for the first time as well as 3DNS Controllers, you'll need to do additional planning. To start, review both this chapter and Chapter 2, *Preparing for Installation*, in the *Administrator Guide for the BIG/ip Controller.*
- If you are preparing to incorporate single network servers or other server array controllers, there may be additional issues to consider, depending on the different products' requirements and configuration.
- Allow access to the necessary ports for communications between 3DNS Controllers, BIG/ip Controllers, and other network equipment. Consult *Port and protocol usage*, on page 2-25 for details.

Planning the primary DNS

As mentioned in Chapter 1, all DNS servers can resolve names, but only primary DNS servers are an authoritative source for zone information.

This section provides examples of name resolution transactions for the following situations:

- The primary DNS is located outside of your network.

- The primary DNS is migrated to a 3DNS Controller. The migration procedure is also provided.

The name resolution process for either situation is similar. The difference is that when the primary DNS is outside of your network, name resolution requests for specified domains are delegated from the primary DNS to the 3DNS Controller. When a 3DNS Controller is the primary DNS, there is no delegation process.

Working with a primary DNS outside of your network

If you're adding 3DNS Controllers into an existing network, you probably have an existing primary DNS in place. Figure 2.1 is an example of the name resolution process where the primary DNS is located outside of the 3DNS network.

The numbers in the illustration correspond to the steps of the process that follows.

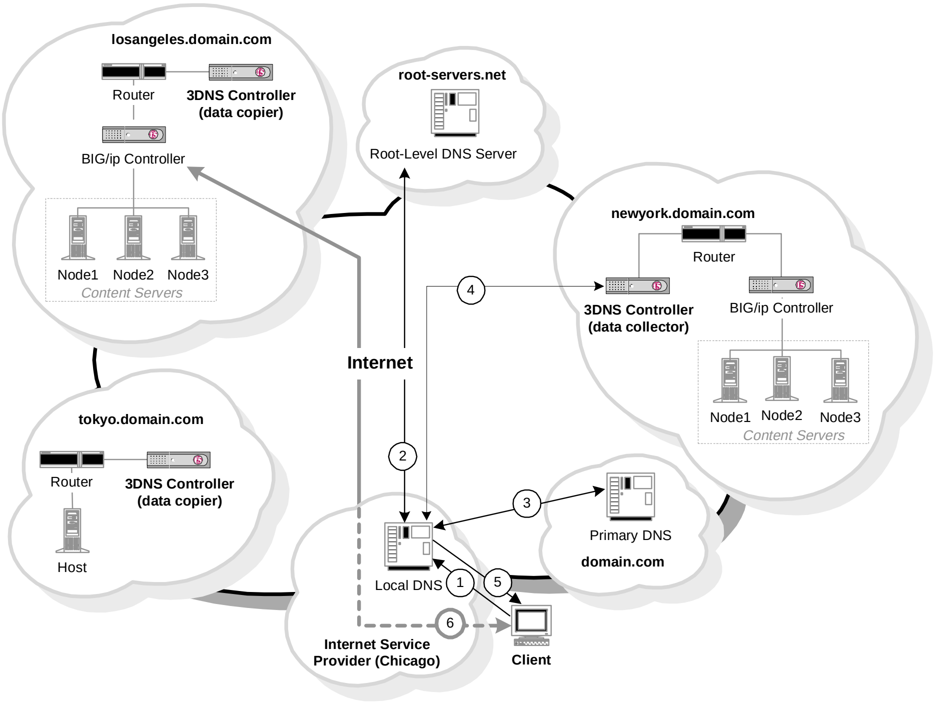


Figure 2.1 *Name resolution process (primary DNS outside of 3DNS network)*

The transaction process is as follows:

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' cache, the ISP's local DNS queries a root server (such as InterNIC's root servers). The root server returns the IP address of a DNS associated with *domain.com*.

3. The ISP's local DNS connects to the primary DNS to resolve *domain.com*. The primary DNS refers the local DNS to the 3DNS Controller in New York because a subdomain was delegated to the 3DNS Controller, making the 3DNS

Controller the authoritative source for this subdomain. The primary DNS created an alias (CNAME) for the domain name to a name in the subdomain that is managed by the 3DNS Controller. This alias is the name that is made public.

4. The local DNS queries the 3DNS Controller in New York for the name resolution, which responds with the IP address to use for the connection.
5. The local DNS passes this IP address back to the client.
6. The client connects to the selected virtual server, which is managed by the BIG/ip Controller in Los Angeles, via the ISP. Note that a portion of the line is dotted to indicate that the actual hardware for this step is not shown, due to the number of ways ISPs can configure their networks.

The choice of data center is based on collected metrics information and load balancing algorithms. This information is not collected during the actual transaction, but at specified intervals. Details on update intervals are given in *Periodic task intervals*, on page 7-8. For details on the available load balancing modes, see Chapter 5, *Load Balancing*.

Migrating the primary DNS to a 3DNS Controller

As mentioned earlier, you can configure a 3DNS Controller to act as the primary DNS for the domains it controls.

To migrate the primary DNS to a 3DNS Controller:

1. If you are migrating from a BIND 4 system to a 3DNS Controller, you must convert the *named.boot* file using the */etc/named-bootconf.pl* Perl script. Run the script by typing the following on the command line:

```
/etc/named-bootconf.pl /etc/named.boot > /etc/named.conf
```

2. Find the primary DNS’ resource records and copy them to a directory of the same name on the 3DNS Controller.

3. Give the old DNS machine's IP address to the 3DNS Controller, or modify all the domains managed by the 3DNS Controller at InterNIC by replacing or adding the IP address to each domain's registration record with a modify domain request.

◆ **Note**

InterNIC changes typically take approximately 24 hours to process and confirm, and another 24 hours to propagate after your configuration becomes active. To avoid outages, always keep a secondary system configured and running during this transition.

Using a 3DNS Controller as your primary DNS

Figure 2.2 shows a typical 3DNS transaction where the primary DNS is located on the 3DNS Controller that is the data collector. The numbers in the illustration correspond to the steps of the process described below.

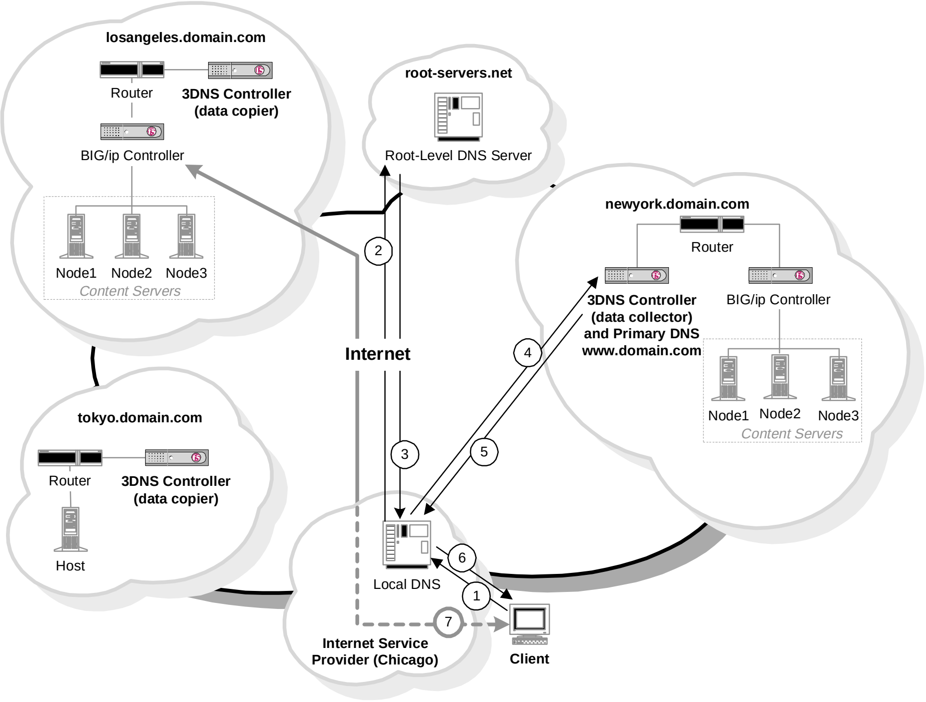


Figure 2.2 *Name resolution process (3DNS Controller as primary DNS)*

The transaction process is similar to that shown in Figure 2.1. The steps in Figure 2.2 are as follows:

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' cache, the ISP's local DNS queries a root server (such as InterNIC's root servers).

3. The root server returns the IP address of a DNS associated with *www.domain.com*.

4. The ISP’s local DNS connects to the primary DNS (in this case, the primary DNS is the 3DNS Controller) for *www.domain.com*. The 3DNS Controller handles the name resolution.
5. The 3DNS Controller responds to the local DNS with the IP address to use for the connection.
6. The local DNS passes this IP address to the client.
7. The client is connected to the selected virtual server, which is managed by the BIG/ip Controller in Los Angeles, via the ISP.

In Figure 2.2, note that part of line 7 is dotted. This is to indicate 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.

Integrating 3DNS Controllers

This section describes issues to consider as you plan which 3DNS Controllers collect data directly from BIG/ip Controllers and hosts, and which 3DNS Controllers simply copy data from the collector 3DNS Controllers. When you are ready to configure data collectors and data copiers, see *Defining data collectors and data copiers*, on page 4-18.

Remember that a ***primary DNS*** is the DNS that is authoritative for zone information. A ***secondary DNS*** can resolve names, but gets its database from a primary DNS. Similarly, a ***data collector*** 3DNS Controller collects metrics information, and a ***data copier*** 3DNS Controller copies metrics from the data collector at specified intervals.

◆ **Note**

Metrics collection occurs independently of name resolution.

Working with a single 3DNS Controller

If you have one 3DNS Controller, you must configure it to be a data collector. As a data collector, it will collect metrics from the BIG/ip Controllers and other host machines on your network. Note that you have the option of defining the 3DNS Controller as the primary DNS.

Working with multiple 3DNS Controllers

When you have more than one 3DNS Controller, you increase the reliability and efficiency of your network. However, you must decide how to handle metrics collection and zone information. For example, suppose you have two 3DNS Controllers, one in New York and one in Los Angeles. The following are some of the ways you can configure these two 3DNS Controllers. Although you can have more than two 3DNS Controllers, the purpose of these examples is to serve as a starting point in the planning process. These examples all assume that the primary DNS is a 3DNS Controller.

Note that the example figures in this section show only how metrics collection is handled, and not the name resolution process. Figures 2.1, on page 2-4, and 2.2, on page 2-7 illustrate the name resolution process.

Example A

Figure 2.3 shows an implementation where both 3DNS Controllers act as primary DNS systems as well as data collectors.

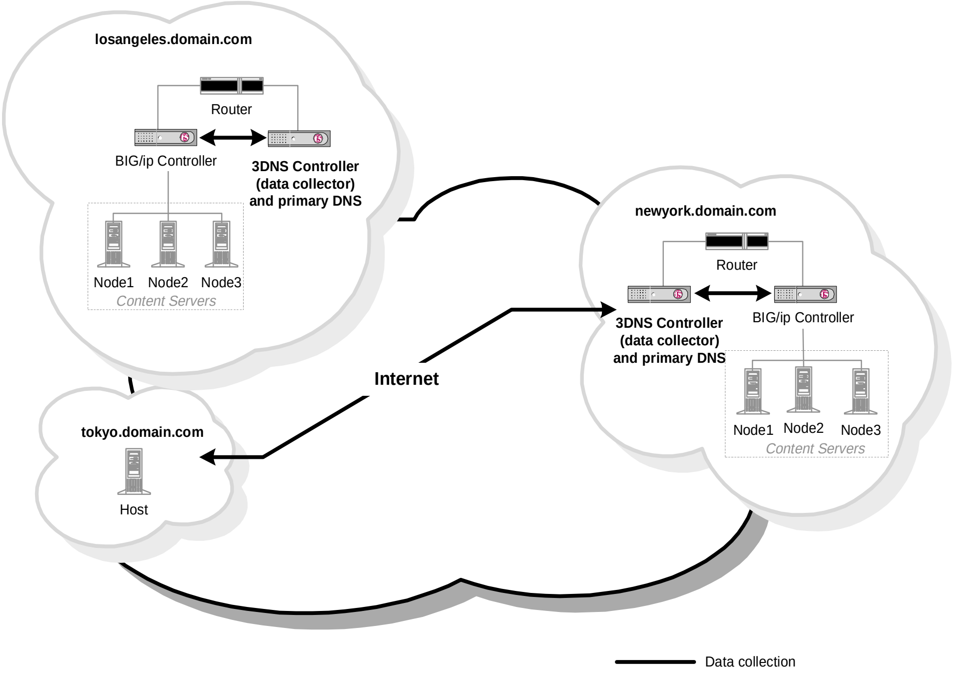


Figure 2.3 *Multiple 3DNS Controllers*

In this case, both 3DNS Controllers perform metrics collection and both are authoritative sources for zone information.

Example B

Figure 2.4 shows an example where the 3DNS Controller in New York is the primary DNS and data collector. The 3DNS Controller in Los Angeles, however, is a secondary DNS and data copier.

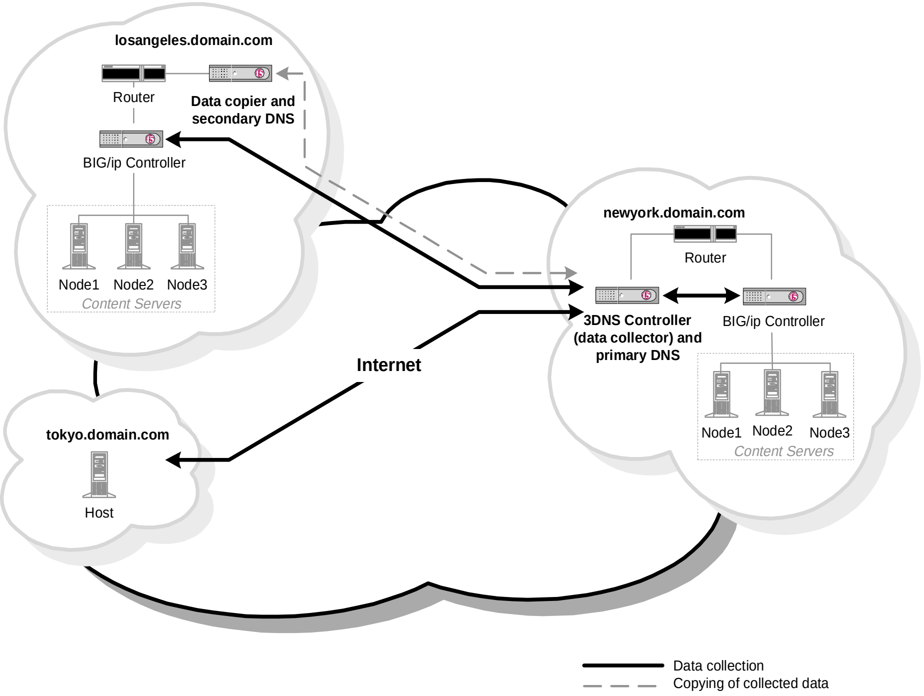


Figure 2.4 *Multiple 3DNS Controllers*

In this case, the 3DNS Controller in New York performs metrics collection. The 3DNS Controller in Los Angeles does not collect metrics, but instead copies metrics from the 3DNS Controller in New York at specified intervals. As in Example A, the 3DNS Controller in New York is the authoritative source for zone information. The 3DNS Controller in Los Angeles is also capable of resolving name requests, but gets its zone information from the New York machine.

Example C

Figure 2.5 shows an example where both 3DNS Controllers are data collectors. The 3DNS Controller in New York is the primary DNS, and the 3DNS Controller in Los Angeles is a secondary DNS.

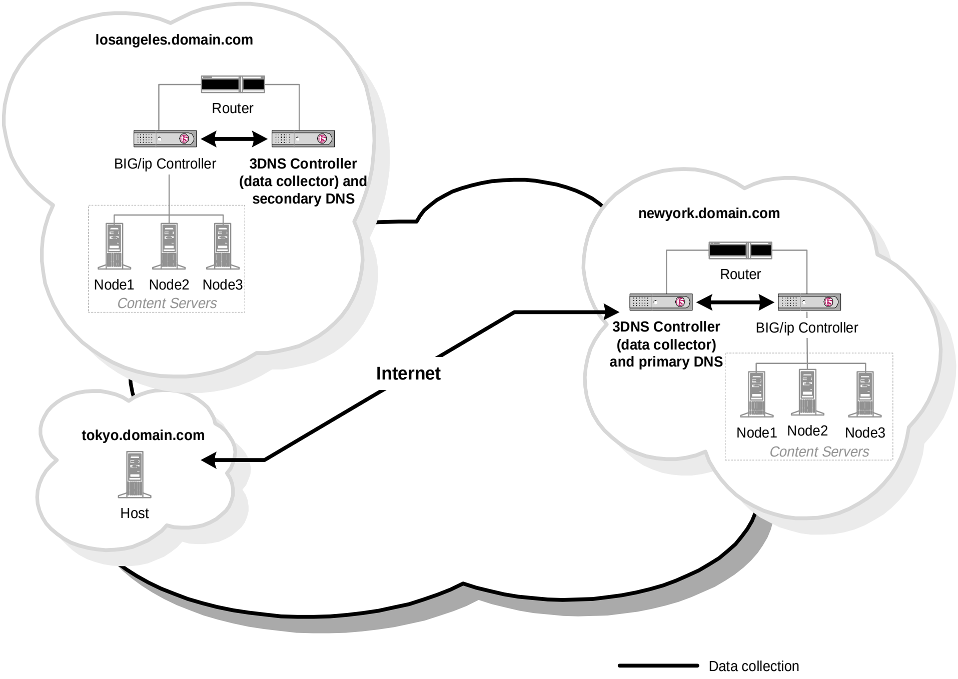


Figure 2.5 *Multiple 3DNS Controllers*

In this case, both 3DNS Controllers perform metrics collection. The 3DNS Controller in New York is the authoritative source for zone information. The 3DNS Controller in Los Angeles is also capable of resolving name requests, but gets its zone information from the New York machine.

Example D

Figure 2.6 shows an example where both 3DNS Controllers are primary DNS systems. The 3DNS Controller in New York is the data collector, and the 3DNS Controller in Los Angeles is a data copier.

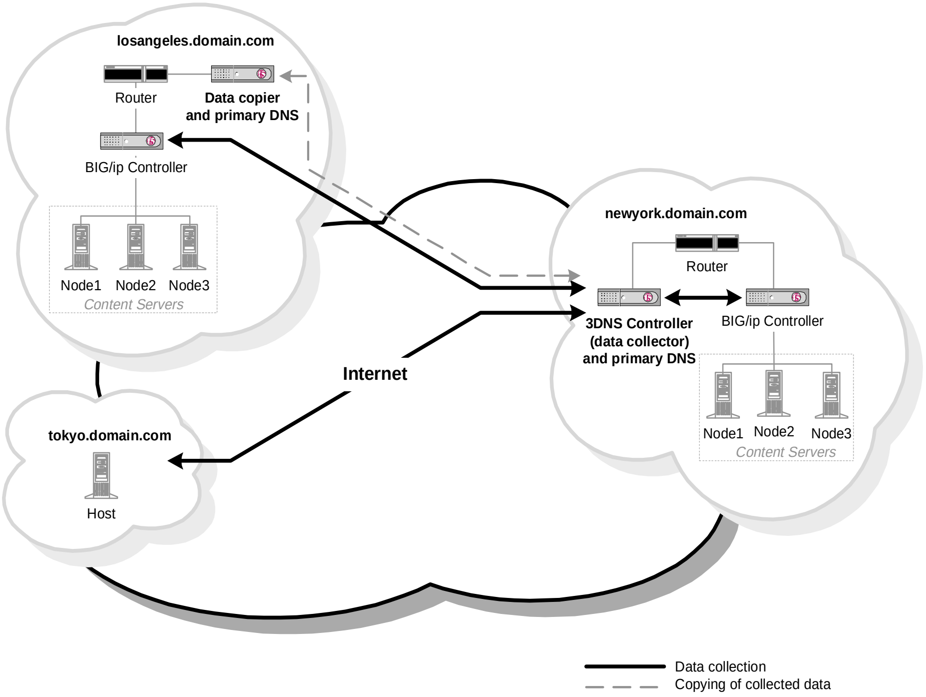


Figure 2.6 *Multiple 3DNS Controllers*

In this case, both 3DNS Controllers are authoritative sources for zone information. The 3DNS Controller in New York is the only machine that collects metrics information.

Advantages and disadvantages

Each configuration example has its advantages and disadvantages. You should evaluate each configuration option carefully before to determine which type of configuration is best suited to your network.

Multiple primary DNS systems

- **Advantages**
  Having more than one primary DNS can be useful in networks where there are a large number of secondary DNS systems. Adding another primary DNS is one possible solution for an overloaded primary DNS.
- **Disadvantages**
  Creating more primary DNS systems creates more work for the administrator. As the administrator, you must synchronize database files between the two systems, or keep track of the differences between each system's zone files.

Secondary DNS

- **Advantages**
  Adding a secondary DNS is the simplest way to add new servers for your domain.
- **Disadvantages**
  An overly large number of secondary DNS systems may overtax the primary DNS. If this is a problem, adding another primary DNS is one possible solution.

Multiple data collectors

- **Advantages**
  Having multiple 3DNS Controllers configured as data collectors adds reliability to your network, because more than one machine has the most current metrics information and can answer queries most intelligently.
- **Disadvantages**
  Having multiple data collectors means that more than one machine is collecting metrics from the BIG/ip Controllers and host machines they manage. This is not a problem unless you

have a large number of data collectors. In this case, the BIG/ip Controllers and host machines may be overloaded having to respond to queries from multiple 3DNS Controller.

Data copiers

- **Advantages**
  Configuring a 3DNS Controller as a data copier can reduce the load on the managed BIG/ip Controllers and host machines, because it reduces the number of queries that the controllers and hosts need to respond to.
- **Disadvantages**
  Data copiers may not have the most current metrics information.

Working with international versions

Using an international version of the 3DNS Controller, the version for use in countries that do not allow encryption, requires additional planning. This section explains how to configure an international 3DNS Controller, and also discusses configuration issues that you must address if you have a mixed environment where international 3DNS Controllers need to communicate with US 3DNS Controllers, and with US and international versions of the BIG/ip Controller and the big3d utility.

Differences between US and international 3DNS Controllers

US 3DNS Controllers are different from international 3DNS Controllers only in the communication tools that they utilize:

- **US 3DNS Controllers**
  US 3DNS Controllers allow secure remote connections via ssh (secure shell), and allow secure copying using scp (secure copy). They also support encryption for iQuery communications between the 3DNS Controller and US big3d utilities that run on BIG/ip Controllers. To allow US 3DNS Controllers to communicate with international 3DNS Controllers, US 3DNS

Controllers include rsh (remote shell) and rcp (remote copy) tools, but they are initially disabled. If you need to configure a US 3DNS Controller to communicate with international 3DNS Controllers, you must explicitly enable the rsh and rcp tools on the US 3DNS Controller. If you need to configure US 3DNS Controllers to communicate with international versions of the big3d utility, you must disable iQuery encryption on US 3DNS Controllers.

- **International 3DNS Controllers**
  International 3DNS Controllers allow remote connections using rsh (remote shell), and allow copying using rcp (remote copy). International 3DNS Controllers do not encrypt iQuery communications between the 3DNS Controller and the big3d utility that runs on BIG/ip Controllers. However, this does not prevent an international 3DNS Controller from successfully making iQuery requests to a US version of the big3d utility.

◆ WARNING

*The **Install and Start big3d** item on the 3DNS Maintenance menu installs the US or international version of the big3d utility depending on whether the 3DNS Controller from which you execute the command is a US version or an international version. In a mixed environment, we recommend that you manually install the appropriate version of the big3d utility on each BIG/ip Controller rather than using the **Install and Start big3d** menu item.*

Configuring international 3DNS Controllers

When you run the First-Time Boot utility to configure an international 3DNS Controller, certain screens are different from those you would normally see if you were running 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.

All other configuration issues are automatically handled by the international 3DNS Controller.

Allowing communications between US and international 3DNS Controllers

There are two situations in which a 3DNS Controller needs to communicate with other 3DNS Controllers: when you synchronize configurations between one 3DNS Controller and another; and when data copiers copy metrics data from a data collector.

If you work in a mixed environment where you have both international and US 3DNS Controllers that need to communicate with each other, you must change the US 3DNS Controller configuration by enabling the remote login tools, including rsh and rcp. You do not need to make any configuration changes to international 3DNS Controllers.

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.

◆ **Note**

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

Allowing communications between international 3DNS Controllers and BIG/ip Controllers

International 3DNS Controllers use rsh and rcp to communicate with BIG/ip Controllers. Note that only BIG/ip Controller version 2.0.1PTF-03 supports rsh and rcp, and that you must explicitly enable these rlogin tools on each BIG/ip Controller that the international 3DNS Controller communicates with, regardless of whether the BIG/ip Controller is a US or an international version.

To enable the rlogin tools on a BIG/ip Controller

1. Use ftp to copy the */usr/contrib/bin/rsetup* file from the 3DNS Controller to */usr/contrib/bin/rsetup* on the BIG/ip Controller.
2. On the BIG/ip Controller, update the permissions in the */usr/contrib/bin/rsetup* file to match the corresponding file permissions as they are set on the 3DNS Controller.
3. 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.

Allowing communications between US 3DNS Controllers and international big3d utilities

US 3DNS Controllers issue encrypted queries to big3d utilities that run on BIG/ip Controllers. In a mixed environment where a 3DNS Controller may have to issue queries to both US and international big3d utilities, you must disable iQuery encryption on the US 3DNS Controller. To disable encryption, set the following global variable to no:

```
encryption no
```

Understanding virtual servers

The 3DNS Controller load balances DNS requests to individual virtual servers. A virtual server is a specific combination of a virtual IP address and a virtual port number.

Virtual servers can be managed by BIG/ip Controllers, or they can be managed by generic host servers, such as a standard network server, a web server, or an array controller. For this reason, the load balancing pools that you define in the 3DNS Controller configuration are broken down into two types:

- **vsb**
  Vsb pools load balance virtual servers associated with BIG/ip Controllers.
- **vsh**
  Vsh pools load balance virtual servers associated with hosts.

These terms, *vsb* and *vsh*, also appear in the Web Administration tool.

◆ **Note**

3DNS Controllers do not collect metrics data or support dynamic load balancing for virtual servers managed by other host machines. However, 3DNS Controllers can perform all static load balancing modes for virtual servers managed by hosts.

The process of configuring virtual servers varies by type:

- **Configuring vsb pools**
  First define each BIG/ip Controller and its virtual servers in a `bigip` statement, and then configure one or more pools in the `wideip` statement using that BIG/ip Controller’s virtual servers.
- **Configuring vsh pools**
  First define each host and its virtual servers in a `host` statement, and then configure one or more pools in the `wideip` statement using that host’s virtual servers.

You may also want to review the following sections for more information:

- *The bigip statement*, on page 7-16. This section provides syntax for adding BIG/ip Controllers and their virtual servers.
- *The host statement*, on page 7-19. This section provides syntax for adding host machines and their virtual servers.
- *Defining a wide IP*, on page 4-5. This section provides a step-by-step guide to configuring wide IPs so that you can perform load balancing.
- *Example syntax for global availability*, on page 5-30. This section provides examples for common load balancing situations.
- Chapter 7, *Statements and Comments*. This chapter provides complete syntax for all statements.
- The *Administrator Guide for the BIG/ip Controller.* Provides information on configuring virtual servers on the BIG/ip Controller.

The iQuery protocol

The iQuery protocol is a UDP-based protocol used to communicate and exchange information between BIG/ip Controllers and 3DNS Controllers. All 3DNS Controllers that are configured as data collectors send queries to BIG/ip Controllers via port 245 or 4353 using the iQuery protocol. You can distribute return iQuery traffic across individual ephemeral ports, or you can use either port 245 or 4353 as a single port for return iQuery traffic. See *Configuring iQuery options*, on page 4-20.

You can enable encryption for iQuery protocol transactions. See *Enabling encryption on US 3DNS Controllers*, on page 4-3. However, if you have a 3DNS Controller in a country that does not allow encryption, see *Working with international versions*, on page 2-15.

Setting up the big3d utility

The big3d utility is the listener that runs on each BIG/ip Controller and 3DNS Controller, and it processes and responds to queries received from data collector 3DNS Controllers.

The big3d utility can be used only with BIG/ip software version 1.8.3 or later. To determine which version of big3d you are using, use the **Check versions of named, BIG/ip kernel and needed big3d** item on the 3DNS Maintenance menu.

To install and run the appropriate version of big3d on each BIG/ip Controller, use the I**nstall and Start big3d** item on the 3DNS Maintenance menu.

big3d configuration options are described in *Configuring the big3d process*, on page D-25.

Understanding probing

Before you install and configure 3DNS Controllers, it is helpful to understand how the probing process works. This section provides an overview of the probing process and an example of a typical sequence of events.

Path probing and the discovery factory

The 3DNS Controller collects a list of the local DNS servers that request name resolutions from the 3DNS Controller. For the purpose of load balancing future connection requests, the 3DNS Controller collects statistics about the paths (such as round trip time and packet completion rate) between each local DNS and each BIG/ip Controller that the 3DNS Controller manages. 3DNS Controller version 1.0.6 improves path statistics collection over older product versions in three ways:

- **Running multiple probing factories**
  Each big3d utility runs multiple probing factories at one time, and can process up to 20 times the number of probe targets than in earlier versions.

- **Dynamic probe protocol switching**
  The big3d utility dynamically switches to the alternate probe protocol (specified by `rtt_probe_dynamic`) in an effort to generate a successful response if the initial probe on a local DNS fails.
- **Implementing the discovery factory**
  The big3d utility supports a discovery factory. If the probing factories fail to get a response from port 53 on a given local DNS using either probe protocol, the 3DNS Controller sends the target local DNS to the discovery factory. The discovery factory scans the target, looking for an open port on which it can receive and respond to a probe. If the discovery factory finds an open port, the 3DNS Controller uses that port for future probes. If it cannot find an open port, the target is no longer probed.

For each requesting local DNS, you can view the current state of probing and discovery in the 3DNS Web Administration tool (see the Local DNS screen). There are six different probe and discovery states as shown in the following table:

State	Description
Needs Probe	Target has never been probed or scanned.
Idle	Target has been successfully probed and is waiting for next probe.
In Probe	Target is currently being probed.
Needs Discovery	Target failed a probe, and now needs to be scanned.
In Discovery	Target is currently being scanned.
Suspended	Target failed the scan and is no longer eligible for probing or scanning.

The following global variables let you control the behavior of the probing and discovery mechanisms, and the way in which the 3DNS Controller uses path data to make load balancing decisions. For information on these variables and all other global variables, see *The globals statement*, on page 7-4.

```
rtt_probe_dynamic
rtt_port_discovery
rtt_discovery_method
rtt_sample_count
rtt_packet_length
rtt_probe_protocol
timer_get_path_data
path_max_refreshes
path_ttl
paths_never_die
paths_noclobber
check_dynamic_depends
```

The probing and discovery process

The following steps outline the typical sequence of events for probing and discovery of a local DNS server.

◆ **Note**

In this example, `rtt_probe_protocol` *is set to* `icmp` *and* `rtt_probe_dynamic` *is set to* `yes`.

1. The 3DNS Controller sends a new set of target local DNS servers to the big3d utility for probing. The more often a target local DNS requests name resolutions from the 3DNS Controller, the more frequently the 3DNS Controller probes the target and refreshes the target's path metrics.
2. The big3d utility begins running the target local DNS servers through its probing factories.
3. For each target local DNS server, the target is first probed using the `rtt_probe_protocol` set by the administrator.

4. If the first probe fails, the big3d utility switches the `rtt_probe_protocol` to the alternate probe protocol, and again probes the target on port 53, this time using the alternate probe protocol.

5. After the big3d utility runs all target local DNS servers through the probing factory, the big3d utility returns the probe results to the 3DNS Controller. The returned metrics include the round trip time, the number of successful replies, and the successful probe protocol.

6. The 3DNS Controller periodically scans the cache for targets that do not have metrics returned from a big3d utility. The 3DNS Controller determines whether probing failed on port 53 for each of these targets. If so, the 3DNS Controller sends the targets to any available big3d for processing in the discovery factory, which determines whether the target has another open port that can be used for probing.

7. For each target local DNS, the big3d discovery factory scans a short list of alternate ports, looking for a response. The port numbers it scans include 21, 22, 23, 25, 80, 110, 113, 139, 248, 1127, 1524, 1525, and 2105. These ports are shuffled before each scan. The discovery factory stops scanning the target upon the first successful response.

8. If the discovery factory fails to get a response from all ports on the short scan list, the discovery factory then scans the target one final time using the ports specified in the */etc/services* file (stored on the machine where that big3d utility resides). You can edit the */etc/services* file to control which ports are scanned when the discovery factory makes a second pass. (Be sure to make a backup copy of the */etc/services* file before you edit it.) Again, note that the port list is shuffled before each scan.

9. After all target local DNS servers have been run through the discovery factory, the big3d utility returns the results back to the 3DNS Controller.

10. If the 3DNS Controller receives a failed target back from the discovery factory, it switches the target local DNS system to the Suspended state. The 3DNS Controller no longer

attempts to probe or scan the target, nor does it use path-related dynamic load balancing modes to resolve requests issued by the local DNS system. If the `preferred` load balancing method is set to a path-related dynamic mode, the 3DNS Controller instead uses a load balancing mode specified by either the `alternate` or the `fallback` load balancing method in the `wideip` statement.

Port and protocol usage

Table 2.1 lists all the ports and protocols used for 3DNS Controller communications.

From	To	Protocol	Port	Purpose
3DNS	BIG/ip	udp	245	iQuery
3DNS	BIG/ip	udp	4353	iQuery (when `use_alternate_iq` = `yes`)
BIG/ip	3DNS	udp	>1024	iQuery
BIG/ip	3DNS	udp	245	iQuery (when `multiplex_iq` = `yes`)
BIG/ip	3DNS	udp	4353	iQuery (when `use_alternate_iq` = `yes` and `multiplex_iq` = `yes`)
Admin	3DNS	tcp	4999	Web administration (HTTP)
3DNS	Admin	tcp	>1024	Web administration (HTTP)
BIG/ip	3DNS	tcp	22	SSH/SCP
3DNS	BIG/ip	tcp	<1023	SSH/SCP
3DNS	BIG/ip	tcp	22	SSH/SCP
BIG/ip	3DNS	tcp	<1023	SSH/SCP
3DNS	3DNS	tcp	22	SSH/SCP
3DNS	3DNS	tcp	<1023	SSH/SCP
BIG/ip	3DNS	tcp	514	RSH/RCP
3DNS	BIG/ip	tcp	>1024	RSH/RCP
3DNS	BIG/ip	tcp	514	RSH/RCP
BIG/ip	3DNS	tcp	>1024	RSH/RCP
3DNS	3DNS	tcp	514	RSH/RCP
3DNS	3DNS	tcp	>1024	RSH/RCP
LDNS	3DNS	udp	53	DNS resolution
3DNS	LDNS	udp	>1024	DNS resolution
LDNS	3DNS	tcp	53	DNS resolution and zone transfers
3DNS	LDNS	tcp	>1024	DNS resolution and zone transfers
BIG/ip	LDNS	icmp		Probing
3DNS	LDNS	icmp		Probing

From	To	Protocol	Port	Purpose
BIG/ip	LDNS	tcp	53	Probing (`rtt_probe_protocol` = `tcp` or `rtt_probe_dynamic` = `yes`) (CISCO routers should "allow establish")
LDNS	BIG/ip	tcp	2000-2300	Probing (`rtt_probe_protocol` = `tcp` or `rtt_probe_dynamic` = `yes`) (CISCO routers should "allow establish")
3DNS	LDNS	tcp	53	Probing (`rtt_probe_protocol` = `tcp` or `rtt_probe_dynamic` = `yes`) (CISCO routers should "allow establish")
LDNS	3DNS	tcp	2000-2300	Probing (`rtt_probe_protocol` = `tcp` or `rtt_probe_dynamic` = `yes`) (CISCO routers should "allow establish")

Table 2.1 *Ports used for 3DNS Controller communications*

Note that you might not need to allow access on all these ports on your network, because you may not need all services. For example, unless you have an international version of 3DNS Controller, you won't use RSH/RCP, which is the only service that requires port 514.

Figure 2.7 shows a subset of the information in the table. For legibility purposes, the specific services are not shown in the figure.

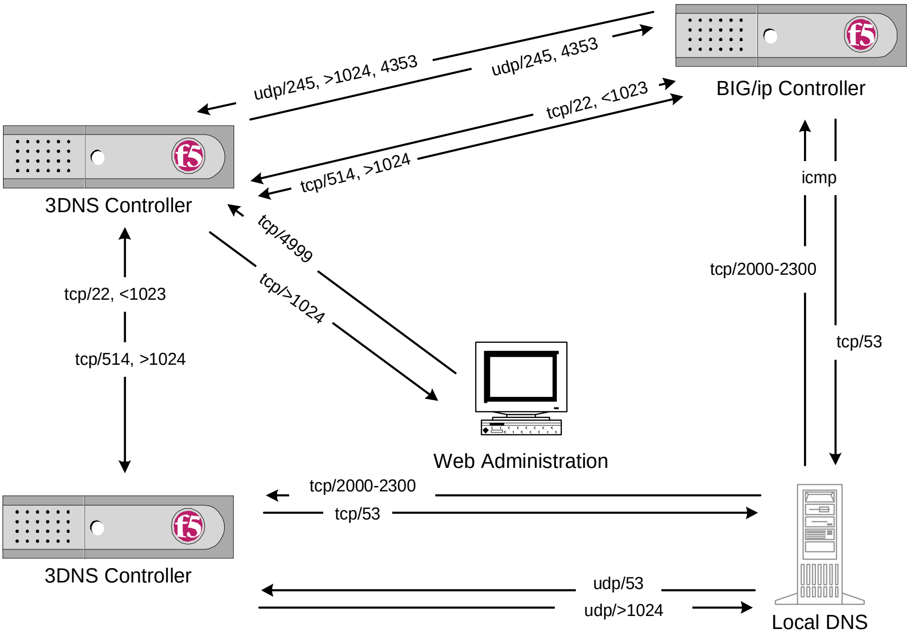


***Figure 2.7** Ports used for 3DNS Controller communications*

3

Installation Procedures

- Installation requirements
- Packing list
- Installation tasks
- The First-Time Boot utility
- F-Secure SSH client
- After installation

Installation requirements

Before you install and use a 3DNS Controller, you must have the following:

- **BIND**
  The primary DNS (which can be a 3DNS Controller) must use BIND, version 4.97 or later. However, we recommend that you use the more current version of BIND, version 8.1.2, or later, that is shipped with 3DNS Controller.
- **Path or route**
  A path or route to each of the BIG/ip Controller's primary or shared interface IP addresses, and to each host.
- **At least one BIG/ip Controller and/or host machine**
  If you plan to use dynamic load balancing, you must have one or more BIG/ip® Controllers running version 1.8.3 or later. You can use static load balancing for host machines or other server array controllers. For information on dynamic and static load balancing modes, see Chapter 5, *Load Balancing*. For information on configuring a BIG/ip Controller, see the *Administrator Guide for the BIG/ip Controller.*

Packing list

When you unpack the 3DNS Controller, check the packing list to ensure that you received all of the following items:

- 3DNS Controller box (1)
- Power cable (1)
- PC/AT-to-PS/2 keyboard adapter (1)
- Keys for the front panel lock (2)
- Extra fan filter (1)
- Rack mounting screws
- *F-Secure SSH User's Guide* (1--US products only)

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:

- You should always install the rack according to the manufacturer's instructions, and be sure to check the rack for stability before placing equipment in it.
- You should 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. Internal temperatures 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.

Installation tasks

The procedures for installation vary depending on whether you are installing a 3DNS Controller for the first time or upgrading an earlier version.

Doing a first-time installation

If you are installing the 3DNS Controller for the first time, you must perform the following tasks:

- **Start the First-Time Boot utility**
  Use the First-Time Boot Utility to install the 3DNS Controller. See page 3-8.
- **Configure F-Secure SSH client**
  You must transfer and install the F-Secure SSH client if you want to be able to configure 3DNS Controllers remotely. See *F-Secure SSH client*, on page 3-14.

Upgrading an earlier version

If you are upgrading from an earlier version of the 3DNS Controller, do the following:

1. Download the *3dns106kit.tar* file from the F5 FTP site:
   ftp://f5dupgrade@ftp.f5.com/3dns/3dns1.0.6
2. Verify the integrity of the file using the `sum` command:
   ```
   sum 3dns106kit.tar
   ```
   If the file is correct, the command displays the correct checksum. Consult the product release notes for the correct checksum value.
3. Extract the *3dns106kit.tar* file in the */var/tmp/* directory:
   ```
   cd /var/tmp
   tar xvf 3dns106kit.tar
   ```

The following table lists the files that are extracted.

File name	Description
3.v1.0.6.tar.gz	3DNS tarball (gzipped)
3dnsbook.pdf	3DNS Controller user manual
backupfile.txt	List of modified configuration files

Again, consult the product release notes for the correct checksum values for each file.

4. Back up the existing configuration files on the 3DNS Controller:

   ```
   cd /var/tmp
   /usr/contrib/bin/gtar -cvf 3dbackup.tar -T
   backupfile.txt
   ```

5. Stop all currently running 3DNS Controller processes:

   ```
   ndc stop
   kill `cat /var/run/big3d.pid`
   kill `cat /var/run/syslog.pid`
   ps -aux|grep thttpd
   kill pid#
   ```

6. Extract the *3.v1.0.6.tar.gz* file in the */var/tmp/* directory:

   ```
   cd /
   /usr/contrib/bin/gtar -zxvpUf
   /var/tmp/3.v1.0.6.tar.gz
   ```

7. Run *3dparse* to update the */etc/wideip.conf* file.

   ```
   3dparse
   ```

8. Restart the 3DNS Controller.

   ```
   sync
   reboot
   ```

Note

Once you install the 3DNS software, you must install new versions of the BIG3d utility on all BIG/ip Controllers managed by the 3DNS Controller. See Setting up the big3d utility, on page 2-21.

Once you install the software update, you must make the required configuration changes described in the following section.

Required configuration changes

The following configuration changes are required. All other configuration changes in this release are optional.

First-Time Boot utility

To check whether the First-Time Boot utility has run, the 3DNS Controller now looks for the */etc/netstart* file rather than */etc/wideip.conf*. If the */etc/netstart* file exists, the 3DNS Controller does not run the First-Time Boot utility at start up. If the 3DNS Controller does not find the */etc/netstart* file, it runs the First-Time Boot utility at start up and saves the */etc/netstart* file upon completion.

Datasize settings

The 3DNS Controller now automatically manages all datasize statements, including process data and stack sizes, based on the amount of memory installed. We recommend that you remove or comment out datasize statements from */etc/named.conf* files because they are no longer necessary.

System control variables on BIG/ip Controllers

If you configure the 3DNS Controller to use the registered iQuery port 4353 for iQuery traffic, you must change the corresponding `bigip.open_3dns_lockdown_ports` sysctl variable on all BIG/ip Controllers running version 2.0 and earlier. The default setting for this variable is 0, but if iQuery traffic is set to run on port 4353, you must change the variable setting to 1.

The big3d utility

All versions of the big3d utility must be updated on BIG/ip Controllers. The 3DNS Controller includes big3d utilities for BIG/ip Controller version 1.8.3, version 2.0, and version 2.0.4. Use the **Install and Start big3d** command on the 3DNS Maintenance menu to automatically copy and install the appropriate version of the big3d utility to all BIG/ip Controllers in your environment.

◆ **Note**

The big3d utility version 2.0.1 is compatible with BIG/ip Controller version 2.0.2.

Storing zone files

Move zone files to the */var/namedb* directory, which offers substantially more storage space than the */etc/namedb* directory.

1. Change the **`directory /etc/namedb`** line in the */etc/named.conf* file to instead point to the */var/namedb* directory:

   ```
   directory /var/namedb
   ```

2. Move */etc/namedb* to */var/namedb.*
3. Restart the *named* process.

Y2K compliance

To make the 3DNS Controller Y2K compliant, you may need to change the serial numbering scheme you apply to zone files. Use the `YYYYMMDDXX` serial number format where the `XX` portion of the number reflects a series number that is attached to the date. This serial number format accommodates zone file transfers that occur more than once in a 24 hour period, but does not create serial numbers that exceed a 32-bit integer. For more information on zone file serial numbers, see page 136 in the O'Reilly & Associates' book *DNS and BIND*, third edition.

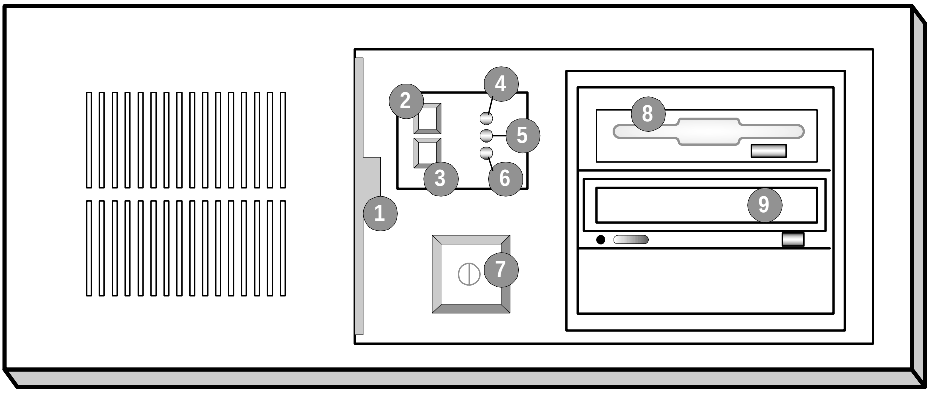
Globals sub-statements

If you are upgrading from an earlier version of 3DNS Controller and you plan to use the RTT or QOS load balancing modes, change the following `globals` sub-statements to the values shown below:

```
paths_noclobber yes
path_ttl 2400
```

The First-Time Boot utility

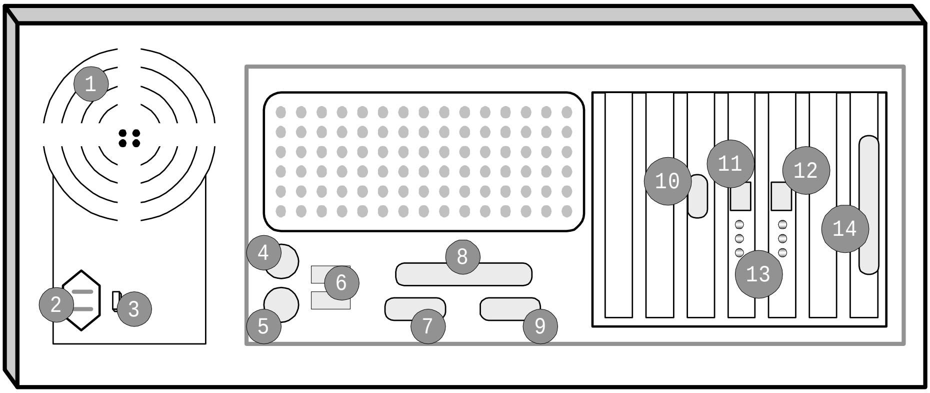
To boot the 3DNS Controller, turn on the power switch located on the front of the 3DNS Controller chassis. The power switch is item 7 on Figure 3.1:



1. Fan filter
2. Keyboard lock
3. Reset button
4. Keyboard lock LED
5. Hard disk drive LED
6. Power LED
7. On/off button
8. 3.5 floppy disk drive
9. CD-ROM drive

Figure 3.1 *3DNS Controller front view*

Figure 3.2 shows the rear of the 3DNS Controller.



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

**Not to be connected to any peripheral hardware.*

Figure 3.2 *3DNS Controller rear view*

When the 3DNS Controller is successfully powered up, you must read and agree to the conditions in the displayed license agreement before the First-Time Boot utility starts and begins prompting you for configuration information.

The configuration is not saved until after you have completely gone through the series of screens. Any changes you need to make to the configuration can be made during the display of the screens to confirm each setting.

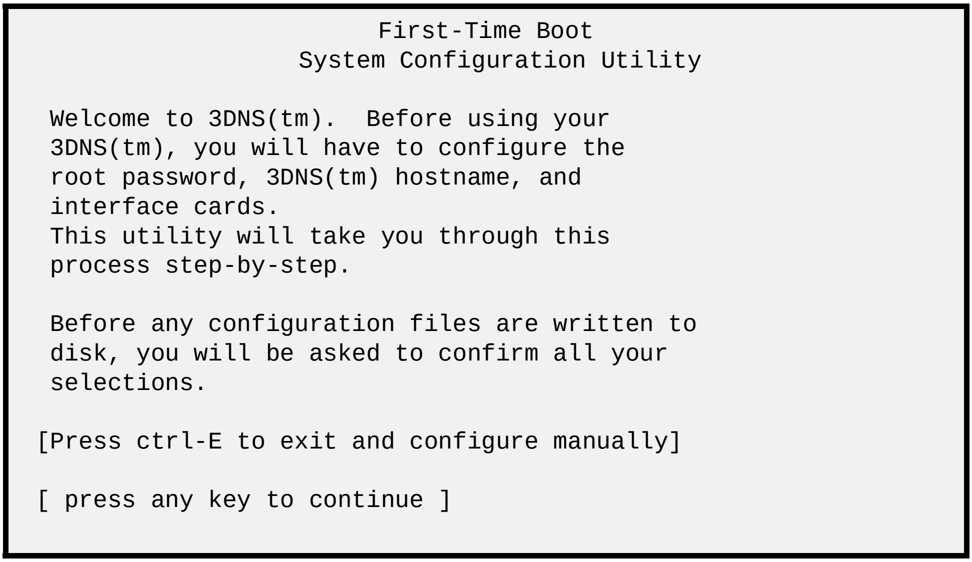
Note

The screens in international versions of 3DNS Controller differ slightly from the screens shown in this section.

Running the First-Time Boot configuration utility

After you press any key at the initial screen, the First-Time Boot Utility screen is displayed, as shown in Figure 3.3.

To continue with the configuration, press any key.



```
                         First-Time Boot
                    System Configuration Utility

   Welcome to 3DNS(tm).  Before using your
   3DNS(tm), you will have to configure the
   root password, 3DNS(tm) hostname, and
   interface cards.
   This utility will take you through this
   process step-by-step.

   Before any configuration files are written to
   disk, you will be asked to confirm all your
   selections.

  [Press ctrl-E to exit and configure manually]

  [ press any key to continue ]
```

Figure 3.3 *First-Time Boot Utility*

Entering the password

At the Set Root Password screen, enter the password that you want to assign to the root user account. The password should be a minimum of six characters, a maximum of 128, and should contain a combination of uppercase, lowercase, and punctuation characters.

Next you are prompted to reset the root password. Press any key to continue.

Confirm password

You are prompted to confirm your new password by typing it again at the second Set Root Password screen. Press any key to continue.

Entering the host name

Enter a fully qualified domain name for the 3DNS Controller (for example, **`3dns.seattle.domain.com`**), and press Enter.

◆ **Note**

If you need to change the host name later, edit the **`hostname <name>`** *line in the* ***/etc/netstart*** *script.*

Setting the interface for the network

In the next series of screens, you set and configure the interface and netmask. To select the interface as either *exp0*, *de0*, or *fddi0*, move the cursor to highlight your selection, and press Enter.

◆ **Note**

The 3DNS Controller First-Time Boot utility lists only the network interface devices that it detects during boot up.

Configuring the interface

Enter the IP address for the interface used in configuration.

Entering a netmask

In this screen you can either accept the default netmask (255.255.255.255), or you can define a custom netmask for the interface.

Enter a broadcast

In this screen you can either accept the default broadcast address (the combination of the IP address and the netmask), or you can define a custom broadcast address for the interface.

Select interface media type

Move the cursor to highlight the media type to be used for the interface, then press Enter. The options for the Interface Media Type are dependent on the NIC being used. An example of media type is as follows:

- auto
- 10baseT
- 10baseT,FDX
- 100baseTX
- 100baseTX,FDX

Setting the remote administrative IP address

Enter the IP address from which you want to perform all remote configuration, administration, and monitoring tasks. Note that you can use an asterisk (*) as a wildcard to specify a range of IP addresses.

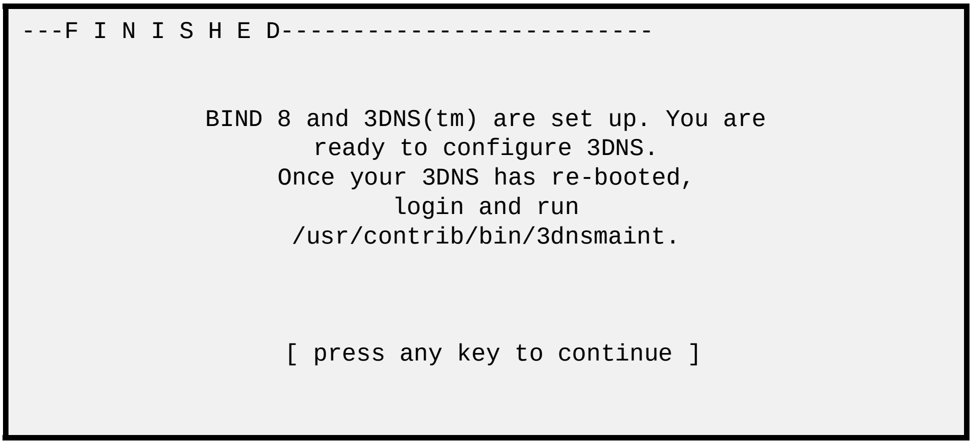
For 3DNS Controllers distributed in the US, administrative command line tasks are conducted using the F-Secure SSH client, which is a secure shell. For international 3DNS Controllers, administrative command line tasks are conducted via Telnet.

Configuring the default route

The default route is used to determine where the 3DNS Controller should send network traffic for which it does not have a static route. The default route is usually the IP address of a router.

Writing the configuration to disk

After you confirm all of your configuration entries, the Finished screen opens, as shown in Figure 3.4.



```
---F I N I S H E D--------------------------


             BIND 8 and 3DNS(tm) are set up. You are
                   ready to configure 3DNS.
                 Once your 3DNS has re-booted,
                       login and run
                /usr/contrib/bin/3dnsmaint.



                [ press any key to continue ]
```

Figure 3.4 *Finished screen*

At this point, the 3DNS Controller writes your configuration to the disk. A status window shows the progress as each of the listed configuration files are saved.

Rebooting the system

Once the First-Time Boot utility is done, press any key to start the 3DNS Controller. At the login prompt, log in as root and halt the system using the **`halt`** command.

After the system halts, set the power switch to the Off position. You must completely power down the 3DNS Controller before attaching it to a network, as described in the next section.

F-Secure SSH client

This section applies only to products sold in the U.S.

If you want to configure the 3DNS Controller from a remote workstation, you need to install the F-Secure SSH client on your remote administration workstation. Note that you can also use the F-Secure SSH suite for file transfer to and from the 3DNS Controller, as well as for remote backups. A F-Secure SSH client is pre-installed on the 3DNS Controller hardware to assist with file transfer activities. Please refer to the F-Secure SSH *User's Guide* shipped with your 3DNS Controller for more information about the SSH client itself.

The F-Secure server is started upon 3DNS Controller boot up. The 3DNS First-Time Boot Utility configures the F-Secure SSH server based on information you provide, so no further modification of the F-Secure configuration is required.

Transferring and installing the F-Secure SSH client

You are licensed to install one (1) copy of the client on your administration workstation. To ease the ordering and installation process, both UNIX and Windows versions of F-Secure SSH client are shipped with the 3DNS Controller. Please contact Data Fellows if you need to purchase additional F-Secure SSH clients, or if you need to purchase the Mac version of the SSH client.

◆ **Note**

The following F-Secure SSH client is shown as an example and may not be an accurate reflection of your administration workstation.

To transfer the F-Secure SSH client to the administration workstation:

1. Using the monitor and keyboard or serial terminal already connected to the 3DNS Controller, change to the directory */usr/contrib/fsecure*, where the F-Secure SSH clients are located. List the directory, noting the file name that corresponds to the operating system of your administration workstation.

2. Start FTP by typing:

   ```
   ftp
   ```

3. Enter passive FTP mode by typing:

   ```
   passive
   ```

4. Open a connection to the administration workstation by typing the following command, where **`<ip_address>`** is the IP address of the administration workstation:

   ```
   open <IP_address>
   ```

   The following text is displayed:

   ```
   Connected to big.f5.com.
   220 big.f5.com FTP server (OSF/1 Version
   5.60) ready.
   Name (big:the user):
   331 Password required for the user.
   Password:
   ```

5. Type your user name and password to complete the connection.

6. Change the transfer mode to binary by typing:

   ```
   bin
   ```

7. Change to the directory on the administration workstation where you want to install the F-Secure SSH client.

8. Transfer the F-secure file to the administration workstation by typing the following command, where **`<file_name>`** is the name of the file corresponding to the operating system of your administration workstation:

   ```
   put <file_name>
   ```

9. Quit FTP on 3DNS by typing:

   ```
   quit
   ```

Using UNIX

To install the F-Secure SSH client on the administration workstation:

1. Log on to the administration workstation and change to the directory where you put the F-Secure SSH client tar file.
2. Untar the file and follow the instructions in the file INSTALL (located in the current directory) to build the F-Secure SSH client for your workstation.

The F-Secure SSH client is now installed on your administration workstation. You are now ready to remotely log on to the 3DNS Controller to finish configuration.

If you have any problems building the F-Secure SSH client for the UNIX operating system on your administration workstation, please contact Technical Support at F5 Networks, Inc.

To remotely log on to 3DNS using F-Secure:

1. Open a connection by typing:

   ```
   ssh -l root [3DNS Controller IP address]
   ```

2. The 3DNS Controller prompts you for the password that you set earlier.

After installation

After the 3DNS Controller is installed, you must perform several configuration tasks to implement the system. These tasks are described in Chapter 4, *Configuring a 3DNS Controller*.

4

Configuring a 3DNS Controller

- **Configuration overview**
- **Configuration tasks**
- **Reference material**

Configuration overview

This chapter describes required and optional tasks for configuring 3DNS Controllers and provides relevant reference material. Another good source of configuration information is Appendix C, *The wideip.conf File*, which provides a sample *wideip.conf* file.

Configuration tasks

Section	Start page
Enabling encryption	4-3
Adding big3d to a BIG/ip Controller	4-5
Adding a wide IP	4-5
Defining data collectors and data copiers	4-18
Configuring iQuery options	4-20

Reference material

Section	Start page
The 3DNS Maintenance menu	4-23
Understanding the wide IP key	4-28
Understanding TTL values	4-28
Troubleshooting configuration problems	4-31

Configuration tasks

As part of setting up a 3DNS Controller, you must do the following:

1. Enable encryption and generate an encryption key. This step is optional, but strongly recommended. See page 4-3.

Note that some countries do not allow encryption. An international version of the 3DNS Controller is available for use in these situations. See *Working with international versions*, on page 2-15.

2. Add big3d to your BIG/ip Controllers. See page 4-5.
3. Add a wide IP. See page 4-5.

   This task requires that you edit the `bigip` and `wideip` statements in your 3DNS Controller configuration file to include the appropriate addresses on your network. You must also edit the `host` statement if you use other hosts on your network. General defaults for the `globals` statement have been implemented, so you don't need to add or edit the `globals` statement unless you want to specify non-default values.
4. Define at least one 3DNS Controller as a data collector and configure the remaining systems as data copiers. See page 4-18.
5. Configure iQuery options. This step is only necessary if you want to specify a non-default port for iQuery traffic or allow for iQuery traffic to pass through firewalls. See page 4-20.

◆ **Note**

*The following information assumes you have read O'Reilly & Associates' book **DNS and BIND** (second or third edition). You can purchase this book from a technical bookstore.*

Enabling encryption on US 3DNS Controllers

You can make iQuery protocol transactions secure by enabling encryption. 3DNS Controller uses the Blowfish CBC encryption algorithm.

◆ **Note**

Encryption is not allowed in some countries. See Working with international versions, on page 2-15.

To enable encryption

1. Open the */etc/wideip.conf* file and change the encryption parameter setting to `yes` (the default setting is `no`). Note that `encryption_key_file` is a string that identifies the name and location of the iQuery key file.

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

2. Open the 3DNS Maintenance menu by typing the following from */usr/contrib/bin*:

   ```
   3dnsmaint
   ```

3. From the menu, select **Generate and Copy F5 iQuery Encryption Key**.

   This command starts the **install_key** script, which creates and distributes the iQuery encryption key to all BIG/ip Controllers and 3DNS Controllers that are currently running big3d utilities.

   For more information, see *install_key and F5makekey*, on page D-26.

Packet validation

An iQuery packet must comply with CRC-32 to be valid. If the packet fails, the 3DNS Controller assumes that the packet is encrypted, and the 3DNS Controller then decrypts and rechecks the packet. If the packet fails CRC-32 once again, the 3DNS Controller logs an error in the syslog facility LOCAL2. You can configure the facility in the */etc/syslog.conf* file.

Adding big3d to a BIG/ip Controller

As described in Chapter 2, big3d is the listener that runs on each BIG/ip Controller and answers 3DNS Controller queries. You must add the big3d utility to each BIG/ip Controller so that the 3DNS Controller can communicate with each BIG/ip Controller.

To add the big3d utility to a BIG/ip Controller:

1. Open the 3DNS Maintenance menu by typing the following command from */usr/contrib/bin*:

   ```
   3dnsmaint
   ```

   The 3DNS Maintenance menu is described on page 4-23.

2. From the menu, select **Install and Start big3d**.

   This starts the *big3d_install* script, which installs the big3d utility on the current BIG/ip Controller.

   You must perform this procedure from each BIG/ip Controller that will be managed by the 3DNS Controller.

   For more information, see *big3d_install*, on page D-24.

Defining a wide IP

You need to define a wide IP statement. Each wide IP statement manages the load balancing of virtual servers on BIG/ip Controllers and other host machines.

A wide IP statement includes the following important information:

- Maps a domain name to a set of virtual servers.
- Assigns a specific load balancing mode to the domain name

◆ **Note**

You can include virtual servers managed by BIG/ip Controllers and other host machines in a single wide IP definition. You can also specify the same host in more than one wide IP definition.

The following instructions include sample wide IP statements that derive from the example configuration introduced in Chapter 2, *Preparing for Installation*. The sample wide IP statement configures a wide IP for the *www.domain.com* domain, where the IP addresses assigned to the 3DNS Controller interfaces are shown in the table below.

3DNS Controller	Interface IP address
New York	192.168.101.2
Los Angeles	192.168.102.2

To add a wide IP

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 E-7.

2. If your network's primary 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:

- If the 3DNS Controller manages the top level for your domain, 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 192.168.101.2
```

3. If your network's primary 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. 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 data center:

Data center	Interface address	BIG/ip or host
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
New York	192.168.105.40	Host

   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

   You configure virtual servers as part of the BIG/ip Controller configuration process. See the BIG/ip Installation and Users Guide for more information.

In the third table, list the other host machines and the IP addresses of the virtual servers that contain the same content. For example:

Host	Virtual server	Virtual port
Tokyo	192.168.104.50	80
New York	192.168.105.50	80

5. 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*.

   See *Understanding the wide IP key*, on page 4-28.

6. 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 4.1** 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 4.2 *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 4.3 *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 4.4 *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
```

7. Configure the `globals`, `bigip`, and `host` statements in */etc/wideip.conf*.

   For the `globals` statement, you need only change parameters if you want to override default values.

   For the `bigip` statements, you must identify each BIG/ip Controller and the virtual servers it owns. In cases where you are using a redundant BIG/ip Controller system, enter the IP address that the redundant system shares between the two units. Do not use the actual address of each BIG/ip Controller in the redundant system.

   For the `host` statement, identify each host machine and its virtual servers.

   Continuing with the example, here are sample `globals`, `bigip`, and `host` statements. Note that each sample is only a snippet of the complete configuration file. For an example of a complete configuration file, see Appendix C, *The wideip.conf File.*

```
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 to tcp probing if icmp
                             // fails
  multiplex_iq yes           // Source port is the same as
                             // destination port for iQuery
  use_alternate_iq_port yes  // Use IANA registered port for
                             // iQuery
}
```

Figure 4.5 *Sample globals statement*

```
bigip {
  // New York
  address 192.168.101.40
  vs {
    address 192.168.101.50
    port 80
    translate {
      address 10.0.0.50
      port 80
    }
  }
}
```

Figure 4.6 *Sample bigip statement*

```
host {
  // Tokyo
  address 192.168.104.40
  vs {
    address 192.168.104.50:80
    probe_protocol  tcp
  }
}
```

Figure 4.7 *Sample host statement*

If you need assistance in defining this section of the file, open the 3DNS Maintenance menu and select **Fetch BIG/ip Configuration**. This menu item starts the *print_3dvips* script, which creates a list of all virtual servers owned by your BIG/ip Controllers. You can use this generated list to enter the correct values for this section of the configuration file. This script is described in *print_3dvips*, on page D-27.

8. Add the *www.wip.domain.com* domain as a wide IP to your *wideip.conf* file. Define which load balancing mode you want to use for the wide IP, and list which virtual servers are to be available for load balancing this wide IP.

   For more information on `wideip` statement syntax, see *The wide IP statement*, on page 7-21.

   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"
  qos_coeff {
    rtt              21
    completion_rate   7
    packet_rate       5
    topology          1
   }

  pool {
    name "pool_1"
    type vsb
    ratio 2
    preferred qos
    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 4.8 *Sample wideip statement*

The wide IP is now in place and configured.

Adding additional wide IPs

After the first wide IP is in place, you can add additional wide IPs. The following procedure assumes that your virtual servers are already defined on the BIG/ip Controllers and other host machines. 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. (For more information on the wide IP key, see page 4-28.)
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 */etc/wideip.conf* 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 // New York
    address 192.168.102.60 ratio 4 // Los Angeles
    address 192.168.103.60 ratio 1 // Tokyo
  }
}
```

Figure 4.9 *Sample wideip statement*

5. Restart the 3DNS Controller by entering the following:

```
ndc restart
```

Defining data collectors and data copiers

When you configure a 3DNS Controller, you configure it as a ***data collector*** or ***data copier***:

- **Data collector**
  A data collector is a 3DNS Controller that collects performance data by issuing queries to big3d utilities that run on BIG/ip Controllers, or on other 3DNS Controllers. The big3d utilities calculate performance data and return the data to the requesting data collector. The data collector stores the performance data in its cache and periodically updates the data.
- **Data copier**
  A data copier is a 3DNS Controller that copies performance data from a data collector. The data copier stores the copied performance data in its cache.

We recommend that you configure the first two 3DNS Controllers in your network to be data collectors, and that you configure any additional 3DNS Controllers as data copiers. For help in planning your network, see *Integrating 3DNS Controllers*, on page 2-8.

Each 3DNS Controller is a data collector until you designate it as a data copier. To designate a 3DNS Controller as a data copier, revise the `globals` statement in its */etc/wideip.conf* file as follows:

```
globals {
   primary_ip <ip_addr>
   sync_db_interval <value>
}
```

The `primary_ip` line defines the IP address of the data collector from which the current data copier copies the performance data. The `sync_db_interval` line sets the frequency at which the data copier queries the data collector for updated performance data.

The above example could be your entire *wideip.conf* file for a data copier, unless you want to set any other global variables to change the behavior of the data copier.

To verify whether a 3DNS Controller is a data collector or data copier, use the Summary screen of the 3DNS Web Administration tool. See *Summary statistics*, on page 6-11.

Synchronizing data copiers

After the data collector is defined, do the following tasks:

- Decide whether to synchronize the *wideip.conf* files on all data collectors. (The *wideip.conf* files on data copiers are short, as shown above.)
- Generate password authentication on each data copier.

Synchronizing *wideip.conf* files

To synchronize the *wideip.conf* files, open the 3DNS Maintenance menu on the 3DNS Controller that is the data collector and select **Synchronize Configuration Data**. This menu item starts the *3dns_sync* script, which distributes the data collector's *wideip.conf* file to all 3DNS Controllers listed in *3dns.txt*.

However, there may be situations where you do not want the *wideip.conf* file to be the same on all 3DNS Controllers. For example, if you are using the Global Availability mode as the default load balancing mode, you need to customize the list of virtual servers in the *wideip.conf* file at each location. Also, remember that the data collector's *wideip.conf* file does not contain the globals sub-statement `primary_ip`. You must add that line to each data copier's *wideip.conf* file.

For more information on synchronizing *wideip.conf* files, see *3dns_sync*, on page D-23.

Generating RSA authentication

To generate RSA authentication, open the 3DNS Maintenance menu on a 3DNS Controller that is a data copier and select **Generate RSA Authentication**. This menu items starts the *3dns_auth* script, which generates password authentication by running the **`ssh-keygen`** command and copying the key to the BIG/ip Controllers and other 3DNS Controllers.

It is important to know that this script only runs **`ssh-keygen`** if no *identity.pub* file exists. An existing *identity.pub* file indicates that **`ssh-keygen`** was already run.

◆ WARNING

Running **`ssh-keygen`** *more than once will cause problems, and is not recommended.*

For more information on password authentication, see *3dns_auth*, on page D-20.

To test that you have successfully generated the ssh key, use ssh to log into the data collector without a password:

```
ssh root@<ip-address-of-3DNS>
```

Configuring iQuery options

You need to configure iQuery options only if you want to specify a non-default port for iQuery traffic, or if you want to allow iQuery traffic to pass through firewalls.

Choosing ports for iQuery traffic

Port 4353 is registered with the IANA as the standard port for the iQuery protocol. You can use the globals sub-statement `use_alternate_iq_port` to specify whether outbound iQuery traffic runs on port 4353, or on port 245. Port 245 is used in earlier versions of 3DNS Controller and is the current default (in order to support backward compatibility). However, we recommend that you set `use_alternate_iq_port` to `yes`, which specifies that the configuration uses the new standard iQuery port, 4353.

◆ Note

If you use port 4353 for iQuery traffic, you must set the corresponding `bigip.open_3dns_lockdown_ports` *sysctl variable to* `1` *(the default setting is* `0`*) on all BIG/ip Controllers running version 2.0 and earlier.*

The 3DNS Controller supports another global sub-statement associated with iQuery traffic. The `multiplex_iq` sub-statement determines whether 3DNS Controller allows all returning iQuery traffic to run only on port 4353 or port 253 (depending on the `use_alternate_iq_port` setting), or allows returning iQuery traffic to run on individual ephemeral ports. The default setting for this variable is `no`, which specifies that returning iQuery traffic runs on individual ephemeral ports.

◆ **Note**

You cannot run the big3d utility on the 3DNS Controller to manage path probing on behalf of hosts if you also want returning iQuery traffic to use a single port. The returning iQuery traffic and the big3d utility create a conflict because they both need to use the same port. To resolve this problem, you should set each host to use a prober than runs on a BIG/ip Controller, rather than on the 3DNS Controller.

Setting up iQuery communications to allow passing through firewalls

The iQuery utility collects configuration and metric information from BIG/ip Controllers on behalf of the 3DNS Controller. The payload information of an iQuery packet contains information that potentially requires translation when there is an intermediate system in the path between a BIG/ip Controller and the 3DNS Controller. In previous versions of 3DNS Controller, iQuery messages included only the configured virtual server address, which was not appropriate where iQuery packets traveled through a firewall and required both the configured address and the translated address. 3DNS Controller now allows iQuery packets to contain both addresses.

In the example configuration shown in Figure 4.10, a firewall separates the path between the BIG/ip Controller 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 state.

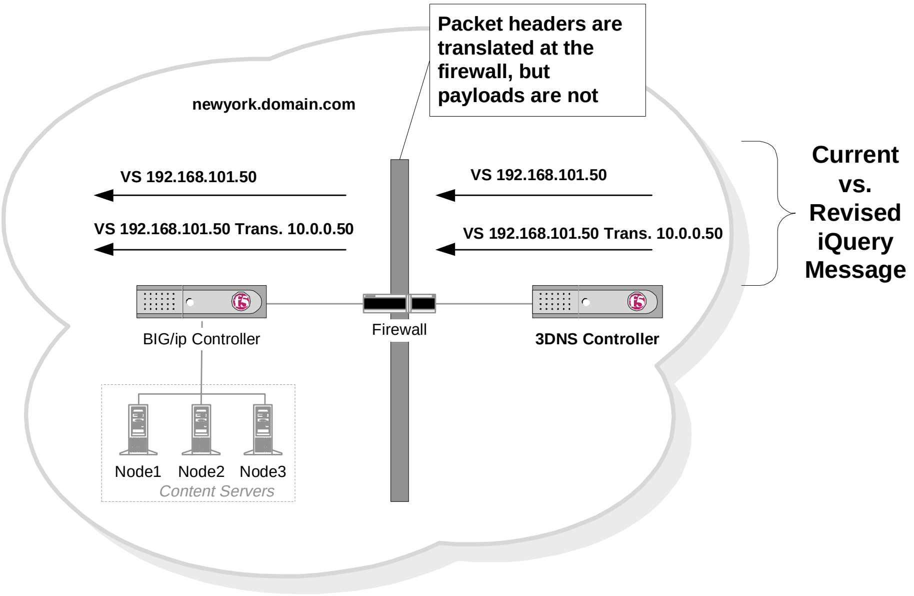

Packet headers are translated at the firewall, but payloads are not
newyork.domain.com
VS 192.168.101.50
VS 192.168.101.50 Trans. 10.0.0.50
VS 192.168.101.50
VS 192.168.101.50 Trans. 10.0.0.50
Current vs. Revised iQuery Message
BIG/ip Controller
Firewall
3DNS Controller
Node1
Node2
Node3
Content Servers

Figure 4.10 *Translating packet address the firewall*

To allow iQuery packets to pass through firewalls, your `bigip` sub-statement needs to include the `translate` keyword. When you include the `translate` keyword, the iQuery utility includes translated IP addresses in the packets sent to the specific BIG/ip Controller.

Here is an example of the appropriate syntax for iQuery firewall translation:

```
bigip {
  address 192.168.101.40
  vs {
    address 192.168.101.50
    port 80
    translate {
      address 10.0.0.50
      port 80
      }
   }
}
```

Reference material

This section describes the 3DNS Maintenance menu (a configuration tool), and background information that is useful in configuring 3DNS Controllers.

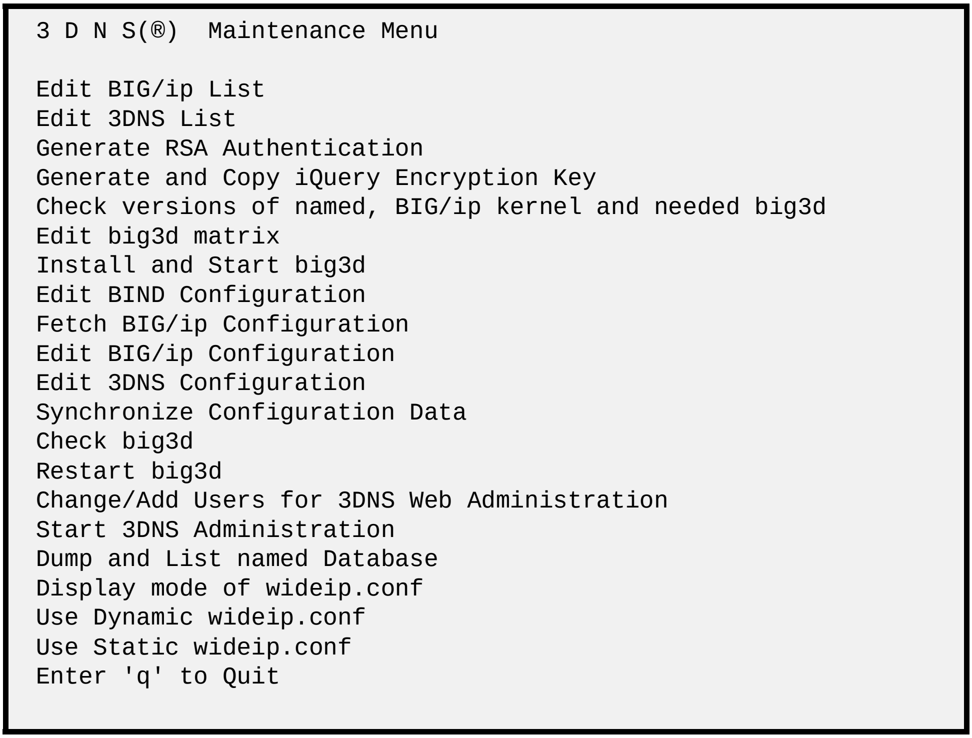
The 3DNS Maintenance menu

You can use the 3DNS Maintenance menu to simplify certain tasks such as starting the big3d utility and distributing the *wideip.conf* file. Many of the menu items correspond to 3DNS Controller scripts; each 3DNS Controller script is described in more detail in Appendix D, *Utilities and Scripts*.

To start the 3DNS Maintenance menu, enter the following command:

`3dnsmaint`

Figure 4.11 shows the 3DNS Maintenance menu:



```
3 D N S(®)  Maintenance Menu

Edit BIG/ip List
Edit 3DNS List
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
Fetch BIG/ip Configuration
Edit BIG/ip Configuration
Edit 3DNS Configuration
Synchronize Configuration Data
Check big3d
Restart big3d
Change/Add Users for 3DNS Web Administration
Start 3DNS Administration
Dump and List named Database
Display mode of wideip.conf
Use Dynamic wideip.conf
Use Static wideip.conf
Enter 'q' to Quit
```

Figure 4.11 *3DNS Maintenance menu*

The following table describes the function of each menu item.

Menu Item	Description
Edit BIG/ip List	Opens the *bigips.txt* data file for editing. For more information on this file, see *File location*, on page D-20.
Edit 3DNS List	Opens the *3dns.txt* data file for editing. For more information on this file, see *File location*, on page D-20.
Generate RSA Authentication	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. This item is not available in the international version of 3DNS Controller.
Generate and Copy F5 iQuery Encryption Key	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. This item is not available in the international version of 3DNS Controller.
Check versions of named, BIG/ip kernel and needed big3d	Displays version numbers for all BIG/ip Controllers known to the 3DNS Controller, as well as the version numbers of the big3d and *named* utilities running on each BIG/ip Controller.
Edit big3d matrix	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 and *named* utilities 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, a new version of big3d must be placed on all BIG/ips Controllers. The **big3d_install** command uses the matrix file to determine which version of big3d to transfer.
Install and Start big3d	Runs the *big3d_install* script, which installs and starts the appropriate version of the big3d utility on each BIG/ip Controller.
Edit BIND Configuration	Opens the *named*.conf file for editing.
Fetch BIG/ip Configuration	Runs the *print_3dvips* script, which reads the list of defined BIG/ip Controllers in the *bigips.txt* file, then retrieves and saves a list of all the virtual servers owned by the listed BIG/ip Controllers. The generated list is saved in a file called */etc/bigip.lst*, and is useful in configuring the `bigip` statement in your *wideip.conf* file.

Menu Item	**Description**
Edit BIG/ip Configuration	Opens the */etc/bigip.lst* file, which is generated by running the *print_3dvips* script (see the preceding description of the **Fetch BIG/ip Configuration** menu item). The */etc/bigip.lst* file contains a list of all the virtual servers owned by the BIG/ip Controllers. Use this menu item to make changes to the `bigip` statement of your *wideip.conf* file: edit the *bigip.lst* file, and then copy and paste it into your *wideip.conf* file.
Edit 3DNS Configuration	Runs the *edit_wideip* script, which opens the *wideip.conf* file for editing.
Synchronize Configuration Data	Runs the *3dns_sync* script, which distributes the *wideip.conf* file from the current 3DNS Controller to all other 3DNS Controllers that are listed in the *3dns.txt* file. Only use the script if you are certain that you want the same *wideip.conf* on all machines. Having the same *wideip.conf* on all machines may not be desirable in all cases.
Check big3d	Runs the *big3d_check* script, which checks that each BIG/ip Controller listed in the *bigips.txt* file is running the big3d utility.
Restart big3d	Runs the *big3d_restart* script, which stops and restarts the big3d utility on each BIG/ip Controller listed in the *bigips.txt* file.
Change/Add Users for 3DNS Web Administration	Runs the *3dns_web_passwd* script, which lets you provide restricted or administrative access to the 3DNS Web Administration site 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 Administration site. If you don't use this script, all users have access to the 3DNS Web Administration site.
Start 3DNS Administration	Runs the *3dns_admin_start* script, which starts the 3DNS Web Administration tool.

Menu Item	Description
Dump and List named Database	Lets you view seven different 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.
Display mode of wideip.conf	Displays the current *wideip.conf* mode: Initial, Static, or Dynamic. Corresponds to the *3dns_mode* script.

Menu Item	Description
Use Dynamic wideip.conf	Creates a static copy of the original *wideip.conf* file, and also creates a dynamic copy of the *wideip.conf* file that includes the path and local DNS data, as well as changes you make using the **Edit wideip.conf** feature in the 3DNS Web Administration tool. Corresponds to the *dynamic _wideip* script. See *Working with static and dynamic wideip.conf files*, on page C-2.
Use Static wideip.conf	Returns to a single *wideip.conf* file, using the *wideip.conf.static* version created when you originally switched the mode to Dynamic. Corresponds to the *static_wideip* script. See *Working with static and dynamic wideip.conf files*, on page C-2.
Enter ’q’ to Quit	Closes the 3DNS Maintenance menu.

Understanding the wide IP key

The wide IP 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.

Understanding TTL variables

Time to Live (TTL) variables control how long information should be saved in the cache and used to make decisions. There are two important TTL values that affect 3DNS Controllers: zone minimums and object limits.

Zone minimums

The zone file contains a Minimum field in the SOA 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 don't want a DNS to cache the answer previously issued 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 7-21.

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 is not saved in any DNS cache. The result is that a new query is made each time a name resolution request is made 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 4.12 *Zone minimums*

Object Limits

Each 3DNS object has an associated TTL. When an object's TTL expires, the 3DNS Controller stops using a dynamic load balancing method and reverts to a static method. You set an object TTL with the `globals` statement. For example:

```
globals {
  bigip_ttl 60
  host_ttl 240
  vs_ttl 120
  path_ttl 600
}
```

Relating 3DNS TTL values to persistence values set on the BIG/ip Controller

You can also configure a TTL value for each wide IP definition. The `ttl` value in a `wideip` statement specifies the amount of time (in seconds) that the specified wide IP's information is to be used by the 3DNS Controller for name resolution and load balancing.

Depending on your situation, you may want to take your configured BIG/ip Controller persistence behavior into account as you configure a wide IP's TTL value.

To find out how a BIG/ip Controller's persistence behavior is configured, check its */etc/rc.sysctl* file. Search for the following line:

```
sysctl -w bigip.persist_time_used_as_limit=
```

The above command ends with a value of either 1 or 0:

- **1**
  Specifies that the persistence time starts when a connection is first made by the client and runs until the persistence time value expires.
- **0**
  Specifies that the persistence timer resets itself upon receipt of each packet. The timer keeps resetting as the client generates traffic over their connection. Once traffic stops on the connection, the timer runs out as the above value.

When you configured your BIG/ip Controller, you specified this behavior using the following command:

```
bigpipe vip <virtual address:port> persist <persistence timeout>
```

If you specified 1 for the above command, configure the corresponding `wideip` statement so that the `ttl` is at least 10 seconds higher than the BIG/ip Controller's persist value.

If you specified 0 for the above command, set the wide IP’s `ttl` value to the maximum value for which you want client connections to persist.

Troubleshooting 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:

- **BIND syntax**
  If you are not well-versed in BIND syntax, or you need a BIND syntax reference, see one of the following:
  - Appendix D of this manual.
  - The O’Reilly & Associates book, *DNS and BIND*.
  - *http://www.isc.org/bind.html*
- ***wideip.conf* syntax**
  After making changes to *wideip.conf*, use the *3dparse* tool to verify syntax before starting *named*. To use this tool, type **`3dparse`** on the command line. (For details on the *3dparse* tool, see page D-2.) For more information on *wideip.conf*, and to see an example of a *wideip.conf* file, see Appendix C, *The wideip.conf File*.
- ***/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.
- **3DNS Controller administration tool**
  The Web Administration tool, described in Chapter 6, *Web Administration*, is useful in diagnosing problems, as it provides a snapshot of your 3DNS Controller network at any given time.

5

3DNS

Load Balancing

- How does load balancing work?
- Load balancing modes
- Load balancing examples

How does load balancing work?

Load balancing is handled on a per wide IP basis. When you select a load balancing mode for a given wide IP, you specify how 3DNS Controllers determine which virtual servers to use for connections.

To set a mode of load balancing for a given wide IP, edit the corresponding `wideip` statement in your 3DNS Controller configuration file.

You can make global load balancing changes using the `globals` statement in the 3DNS Controller configuration file.

See Chapter 7, *Statements and Comments*, for more information on all statements and sub-statements.

Load balancing modes

There are three types of load balancing modes: static, dynamic, and specialized.

Dynamic

Dynamic load balancing modes rely on the iQuery protocol to collect important performance information such as ***Round Trip Times (RTT)***, which calculates the time a local DNS takes to respond to a ping issued by a BIG/ip Controller, or ***Least Connections***, which calculates the current number of current connections for virtual servers on BIG/ip Controllers.

Static

Static load balancing modes do not require network communications (other than ensuring server availability) and exhibit more predictable load distribution.

Static load balancing modes only use the iQuery protocol to collect server status to determine how connections are made. By incorporating server status into the name resolution and load balancing processes, the 3DNS Controller always sends requests to a live BIG/ip Controller or host machine (assuming that live BIG/ip Controllers or host machines are available). You can change this behavior by setting the globals sub-statement `check_static_depends` to `no`.

Both static and dynamic load balancing modes are available when monitoring virtual servers managed by BIG/ip Controllers. However, when monitoring virtual servers managed by other host machines, only the static load balancing modes are available.

Specialized

Specialized modes include the following:

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

Topology access control and e-commerce go beyond simple load balancing in that they let you fine tune how connections are distributed.

Dynamic modes

Dynamic load balancing modes use the iQuery protocol to collect the information that is used to determine how to direct client requests. When you configure a wide IP for a path-dependent dynamic load balancing mode such as Round Trip Times or Completion Rate, the 3DNS Controller instructs each BIG/ip Controller to collect path metrics for the local DNS. The 3DNS Controller requests path metrics from each BIG/ip Controller the first time a name resolution request is made by the local DNS, and thereafter on a periodic basis.

You can control how often the data is refreshed (the interval between updates) using the globals sub-statements `get_path_data` and `path_ttl`. Path metric collection does not occur during the name resolution process.

The 3DNS platform supports these dynamic load balancing modes:

- Completion Rate
- Least Connections
- Packet Rate
- Round Trip Times (RTT)

Completion Rate
Syntax: `completion_rate`

Figure 5.1 shows the 3DNS Controller using the Completion Rate load balancing mode. The Completion Rate mode selects a virtual server on the BIG/ip Controller which currently maintains the least number of dropped or timed out packets for transactions between itself and the local DNS.

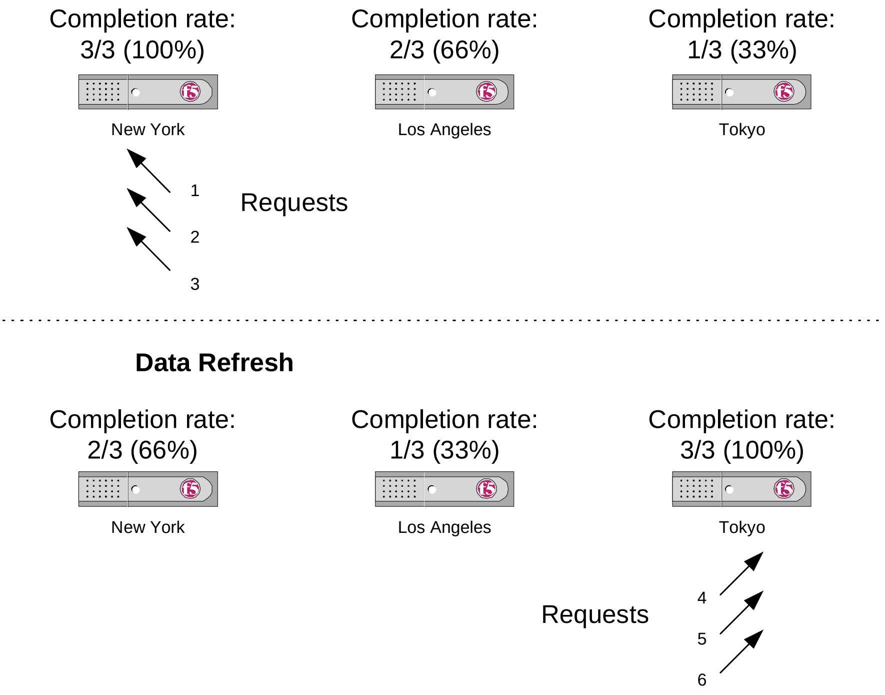


***Figure 5.1** Completion rate mode*

Example syntax

```
// Completion rate
wideip {
  address 192.168.101.60
  port 80
  name "cgi.wip.domain.com"
  pool {
    name "mypool"
    type vsb
    preferred completion_rate
    address 192.168.101.60 // New York
    address 192.168.102.60 // Los Angeles
    address 192.168.103.60 // Tokyo
  }
}
```

Figure 5.2 *Example syntax for completion rate*

Related globals sub-statements

`timer_get_path_data`

`path_ttl`

`rtt_timeout`

`rtt_sample_count`

`rtt_packet_length`

For information on these and all `globals` sub-statements, see *The globals statement*, on page 7-4.

Least Connections
Syntax: `leastconn`

The Least Connections mode selects a virtual server on the BIG/ip Controller which currently maintains the least number of connections.

Example syntax

```
// Least connections with ratio as an alternate
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 // New York
    address 192.168.102.60 ratio 4 // Los Angeles
    address 192.168.103.60 ratio 1 // Tokyo
  }
}
```

Figure 5.3 *Example syntax for least connections*

Related globals sub-statements

`timer_get_vs_data`

`vs_ttl`

For information on these and all `globals` sub-statements, see *The globals statement*, on page 7-4.

Packet Rate
Syntax: `packet_rate`

Figure 5.4 shows the 3DNS Controller using the Packet Rate load balancing mode. The Packet Rate mode selects a virtual server which corresponds to the BIG/ip Controller that is currently processing the least packets per second.

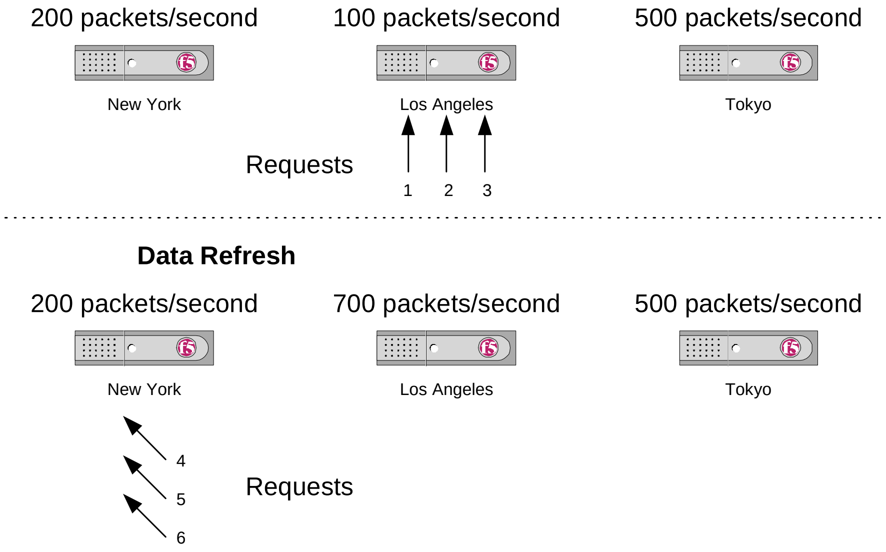


***Figure 5.4** Packet rate mode*

Example syntax

```
// Packet rate
wideip {
  address 192.168.101.60
  port 80
  name "cgi.wip.domain.com"
  pool {
    name "mypool"
    type vsb
    preferred packet_rate
    address 192.168.101.60 // New York
    address 192.168.102.60 // Los Angeles
    address 192.168.103.60 // Tokyo
  }
}
```

***Figure 5.5** Example syntax for packet rate*

Related globals sub-statements

```
timer_get_bigip_data
bigip_ttl
```

For information on these and all `globals` sub-statements, see *The globals statement*, on page 7-4.

Round Trip Times (RTT) Syntax: `rtt`

Figure 5.6 shows the 3DNS Controller using the Round Trip Times load balancing mode. The Round Trip Times (RTT) mode selects the virtual server with the fastest measured round trip time using probes from the BIG/ip Controller to the client's local DNS.

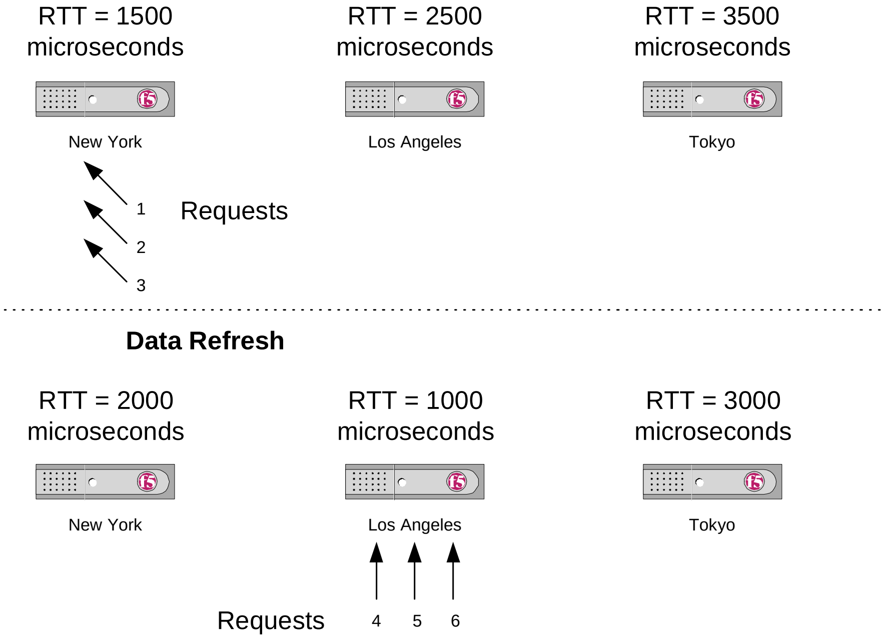


Figure 5.6 *Round Trip Times mode*

In the top half of Figure 5.6, the New York machine has the lowest score. As a result, the 3DNS Controller selects New York for connections until the round trip times are recalculated. After the data was refreshed, the Los Angeles machine had the lowest score, so subsequent requests were sent to Los Angeles.

Example syntax

```
// Round trip time load balancing with topology
// as alternate load balancing
wideip {
  address 192.168.103.60
  port 80
  name "ntp.wip.domain.com"
  pool {
    name "poolA"
    type vsb
    preferred rtt
    alternate topology
    address 192.168.101.60 // New York
    address 192.168.102.60 // Los Angeles
    address 192.168.103.60 // Tokyo
  }
}
```

Figure 5.7 *Example syntax for round trip time*

Related globals sub-statements

`timer_get_path_data`

`path_ttl`

`rtt_timeout`

`rtt_sample_count`

`rtt_packet_length`

For information on these and all `globals` sub-statements, see *The globals statement*, on page 7-4.

Static modes

The 3DNS platform supports these static load balancing modes:

- Random

- Ratio
- Round Robin
- Null
- Return to DNS

Random
Syntax: `random`

When you specify a Random load balancing mode, the 3DNS Controller selects a virtual server for the connection at random from the wide IP set of virtual servers.

Example syntax

```
// Random
wideip {
  address 192.168.101.60
  port 80
  name "cgi.wip.domain.com"
  pool {
    name "mypool"
    type vsb
    preferred random
    address 192.168.101.60 // New York
    address 192.168.102.60 // Los Angeles
    address 192.168.103.60 // Tokyo
  }
}
```

Figure 5.8 *Example syntax for random*

Ratio
Syntax: `ratio`

Figure 5.9 shows the 3DNS Controller using the Ratio load balancing mode. The Ratio mode, also known as Weighted or Administrative Cost, is useful for sites that have servers of varying capabilities. You specify what proportion of connections should go to each virtual server. Over a long period of time, the number of requests resolved to each virtual server in the set is in proportion to the specified weights. This load balancing mode is similar to Round Robin, but with weights assigned to each server. The default ratio for all servers is 1.

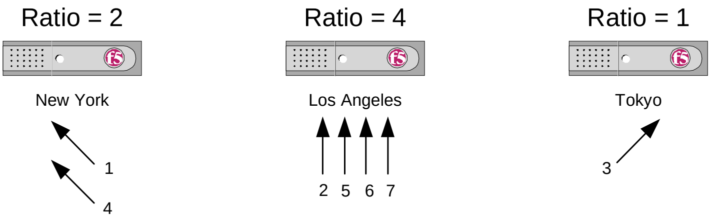


Figure 5.9 *Ratio mode*

Example syntax

```
// Least connections with ratio as an alternate
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 // New York
    address 192.168.102.60 ratio 4 // Los Angeles
    address 192.168.103.60 ratio 1 // Tokyo
  }
}
```

Figure 5.10 *Example syntax for ratio*

Round Robin
Syntax: `rr`, `round_robin`

Figure 5.11 shows the 3DNS Controller using the Round Robin load balancing mode. The Round Robin mode distributes client requests in a circular and sequential pattern. Over a long period of time, the total number of connections for each virtual server is the same.

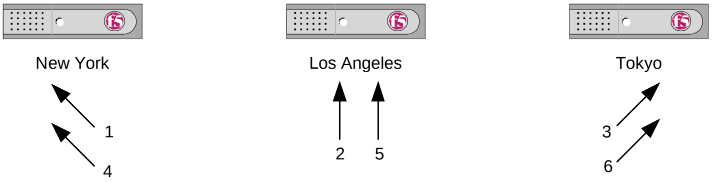


Figure 5.11 *Round Robin mode*

Example syntax

```
wideip {
  address 192.168.102.60
  service "ftp"
  name "ftp.wip.domain.com"
  pool {
    name "main_pool"
    type vsb
    preferred rr
    address 192.168.101.60 // New York
    address 192.168.102.60 // Los Angeles
    address 192.168.103.60 // Tokyo
  }
}
```

Figure 5.12 *Example syntax for round robin*

Null
Syntax: `null`

Specifying the null load balancing mode causes the 3DNS Controller to bypass the current load balancing method. It forces the 3DNS Controller to use the next load balancing method, or to move on to the next available pool.

Return to DNS
Syntax: `return_to_dns`

The return to DNS mode returns the resolution request to DNS, preventing the 3DNS Controller from using the next load balancing method, or using the next available pool.

The following example shows both null and return to DNS.

Example syntax

```
// Global availability pool load balancing between
// bigip data centers 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 "Tokyo"
    type vsb
    ratio 1
    preferred leastconn
    alternate null
    fallback return_to_dns
    address 192.168.103.50 ratio 3
    address 192.168.103.60 ratio 2
    address 192.168.103.70 ratio 1
  }
}
```

Figure 5.13 *Example syntax for null and return to DNS*

Specialized modes

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

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

Topology access control and e-commerce go beyond simple load balancing in that they let you fine tune how connections are distributed.

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.

Defining the topology statement

You insert the `topology` statement at the end of the *wideip.conf* file. The `topology` statement consists of three parameters, `acl_threshold`, `limit_probes`, and `longest_match`, followed by a list of records defining a network.

The syntax is as follows:

```
topology {
    acl_threshold   <1..4294967295>
    limit_probes    <yes | no>
    longest_match   <yes | no>
    <server cidr> <LDNS cidr> <score>
}
```

Figure 5.14 *Syntax for topology statement*

Topology statement example

It is best to explain topology access control with an example. 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 don’t mind if a few requests are sent to Tokyo when requests cannot be resolved in New York or Los Angeles.

However, because of cost issues, you do not want requests made from clients in South America to go to the New York data center. To achieve this, you can configure the topology statement as shown.

```
topology {
  acl_threshold   1
  limit_probes    yes
  longest_match   yes

// Server       LDNS            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

}
```

Understanding the list records

The record list records in the topology statement define a score for pairs of known local DNS servers and data centers. 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 above, say that the local DNS 198.0.0.0 requested name resolution for *www.domain.com*, and a virtual server in the `vsb` 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.

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 following:

```
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
```

Is equivalent to:

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

◆ **Note**

The term list-record (server, local DNS) refers to the longest matching record for the BIG/ip Controller or host IP address, and the local DNS IP address.

Using the longest match rule

The 3DNS Controller uses the same type of longest match rule that is commonly used by routers. 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.

Using wildcard list records to explicitly allow or deny access to local DNS servers that do not match a specific list record

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 `acl_threshold` value and the topology map scores. In the example `topology` statement above, the 3DNS Controller would not send a local DNS 200.0.0.0 connection to the BIG/ip Controller 192.168.101.0 for probing, but would send it to the BIG/ip Controller 192.168.103.0.

Topology load balancing mode
Syntax: `topology`

The topology 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.

Example syntax

```
wideip {
  address 192.168.103.60
  port 80
  name "ntp.wip.domain.com"
  pool {
    name "poolA"
    type vsb
    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 5.15 *Example syntax for topology*

E-commerce

For the purposes of conducting business over the Internet, you can configure the wide IP statement so that connections are not sent to a given address unless all specified ports or services are available. To do so, use the `wideip port_list` sub-statement.

For example:

```
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"
    type vsb
    ratio 2
    preferred qos
    alternate ratio
    address 192.168.101.70 ratio 7
    address 192.168.102.60 ratio 2
  }
```

Figure 5.16 *Syntax for e-commerce*

In the above example, ports 80 and 443 must be available before connections are sent to the specified address. If one of the ports in the list is down, the 3DNS Controller will not send traffic to any of the ports defined in the list.

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 following virtual servers must exist:

```
192.168.101.70:80
192.168.101.70:443
192.168.102.60:80
192.168.102.60:443
```

Use of the `port_list` parameter is not restricted to e-commerce purposes; you can use `port_list` in any situation where you want multiple services to be available for resolving requests.

Quality of Service (QOS)
Syntax: qos

In essence, the QOS mode lets you define a custom load balancing mode. The ***Quality of Service (QOS)*** score is a user-definable metric that includes a configurable combination of the RTT, Completion Rate, Packet Rate, and Topology modes. The virtual server with the highest metric is used for the connection.

Use this equation to configure QOS:

```
A (1/packet rate) + B (1/rtt) + C (completion rate) + D (topology)
```

You specify the coefficients **`A`**, **`B`**, **`C`**, and **`D`**. You can set the coefficients on a global basis. You can also override global values for each wide IP by using a `qos_coeff` declaration in the wide IP definition. The following table shows the user-configurable values that correspond to the coefficients:

Coefficient	Global	Wide IP override for qos_coeff {}
A	qos_coeff_packet_rate	packet_rate
B	qos_coeff_rtt	rtt
C	qos_coeff_completion_rate	completion_rate
D	qos_coeff_topology	topology

The global coefficient settings define default values for all wide IPs that use the QOS load balancing mode. Figure 5.17 shows sample default settings.

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

Figure 5.17 *Global settings*

In a wide IP definition, you can override the global coefficient settings. Figure 5.18 displays a wide IP definition that uses overrides for the global settings shown in Figure 5.17.

```
//
wideip {
  address 192.168.101.50
  service "http"
  name "www.wip.domain.com"
  qos_coeff {
    rtt              21
    completion_rate   7
    packet_rate       5
    topology          1
   }

  pool {
    name "pool_1"
    type vsb
    ratio 2
    preferred qos
    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 5.18 *QOS coefficient settings that override the global default settings*

Balancing QOS coefficients

Before you change QOS coefficients from their default values, note the following:

1. 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.

2. 3DNS Controller normalizes the raw metrics on the order 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.

   The following table lists each 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 times	Microseconds	2,000,000	Lower
Completion rate	0-100%	100%	Higher
Topology	0 (off) or 1 (on)	N/A	N/A

3. 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 round trip times:

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

Figure 5.19 *Balancing QOS coefficients to emphasize round trip time*

In the above example, if round trip times for two virtual servers are close, the virtual server with the best packet rate is chosen. If both round trip times and packet rates are close, the completion rate breaks the tie.

4. If you need help in customizing a QOS equation, contact F5 technical support.

Global Availability
Syntax: `global_availability`, `ga`

Figure 5.20 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.

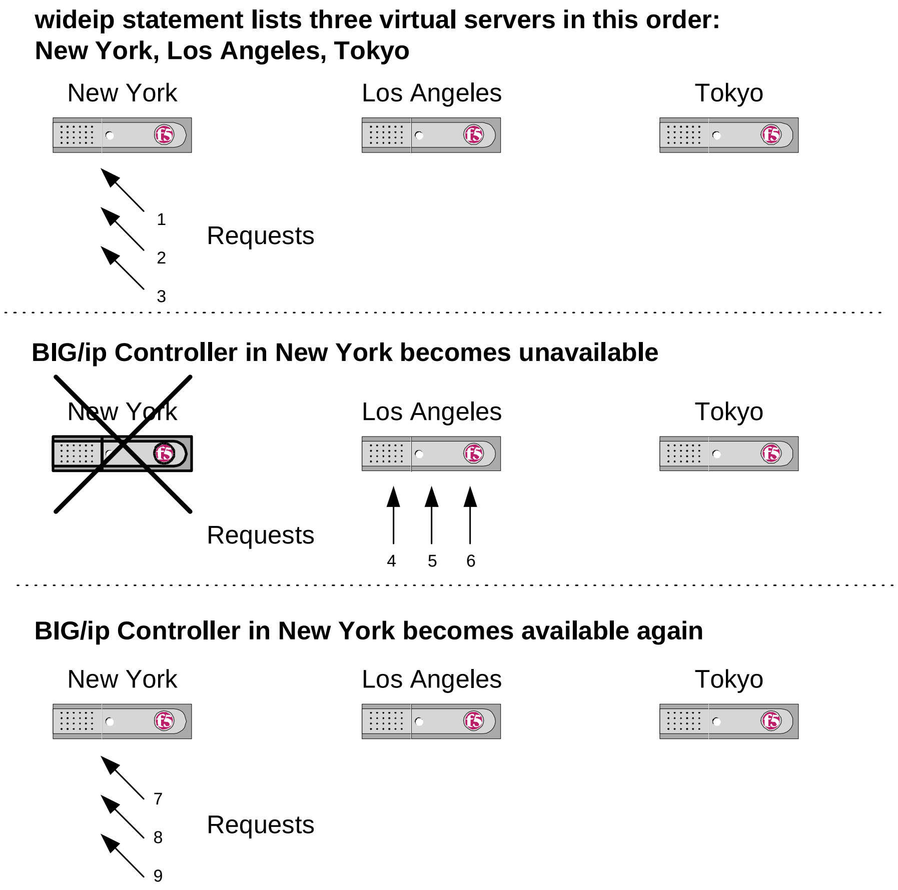

wideip statement lists three virtual servers in this order:
New York, Los Angeles, Tokyo
New York
Los Angeles
Tokyo
1
2
3
Requests
BIG/ip Controller in New York becomes unavailable
New York
Los Angeles
Tokyo
Requests
4
5
6
BIG/ip Controller in New York becomes available again
New York
Los Angeles
Tokyo
7
8
9
Requests

Figure 5.20 *Global Availability mode*

Example syntax

```
// Global availability
wideip {
  address 192.168.101.60
  port 80
  name "cgi.wip.domain.com"
  pool {
    name "mypool"
    type vsb
    preferred ga
    address 192.168.101.60 // New York
    address 192.168.102.60 // Los Angeles
    address 192.168.103.60 // Tokyo
  }
}
```

Figure 5.21 *Example syntax for global availability*

Load balancing examples

The following examples show only a few of the ways different load balancing modes can be used to optimize performance of your network. Use these examples as a starting point for deciding how you want connections handled.

Configuring a standby data center

Using 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.

For example:

```
wideip {
  address 192.168.101.60
  port 80
  name "www.wip.domain.com"
  pool {
    name "pool1"
    type vsb
    preferred ga
    address 192.168.101.60
    address 192.168.102.60
    address 192.168.103.60
  }
}
```

***Figure 5.22** Configuring a standby data center*

Configuring alternate modes

This section provides two examples of how you can use an alternate load balancing method.

Example A

This example uses the Round Trip Times as the preferred mode. If the preferred mode (round trip times) fails, the 3DNS Controller uses the alternate mode. In this example, global availability is the alternate mode. This means that if the preferred mode fails, the 3DNS Controller in this example chooses the first available virtual server from the list in the `wideip` statement.

```
// From the New York wideip.conf
wideip {
  address 192.168.101.60
  port 80
  name "www.wip.domain.com"
  pool {
    name "poolA"
    type vsb
    preferred rtt
    alternate ga
    address 192.168.101.60 // New York
    address 192.168.102.60 // Los Angeles
    address 192.168.103.60 // Tokyo
  }
}
```

Figure 5.23 *Using alternate load balancing modes (New York wideip.conf)*

To cause the 3DNS Controller in New York to fall back to a New York virtual server and the similar effect to take place on the 3DNS Controller in Los Angeles, the virtual servers nearest to the 3DNS Controller are listed first. One unique and important aspect of this type of configuration is that each 3DNS Controller maintains a different *wideip.conf* file, rather than working from a synchronized *wideip.conf* file.

```
// From the Los Angeles wideip.conf
wideip {
  address 192.168.101.60
  port 80
  name "www.wip.domain.com"
  pool {
    name "poolB"
    type vsb
    preferred rtt
    alternate ga
    address 192.168.102.60 // Los Angeles
    address 192.168.101.60 // New York
  }
}
```

Figure 5.24 *Using alternate load balancing modes (Los Angeles wideip.conf)*

Example B

In this example, suppose you are releasing a new version of a software product and plan to distribute it via FTP. You decide to specify the Least Connections as the preferred mode to better guarantee a connection for customers attempting to download the software. You can select Ratio as the alternate mode with weights assigned to certain servers. For this example, assume that the Los Angeles server has the ability to process requests faster than the other servers. Because of this, you choose the Los Angeles server as the preferred virtual server twice as often as the New York server, and three times as often as Tokyo server.

```
// Least connections with ratio as an alternate
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 // New York
    address 192.168.102.60 ratio 4 // Los Angeles
    address 192.168.103.60 ratio 1 // Tokyo
  }
}
```

Figure 5.25 *Using alternate load balancing modes*

Using multiple resource pools

To help you address common issues, this section provides several examples that offer a variety of solutions.

Example A

In this example, suppose you are a network administrator of a large network. Your network includes a large number of Sendmail servers to help distribute the mail traffic. For example, you’ve configured one Sendmail server to serve the human resources department, another to serve the engineering group, and another to serve the sales staff. Some Sendmail servers are virtual servers managed by BIG/ip Controllers, while others are virtual servers managed by a host machine.

Maintaining this configuration is time-consuming, since you must configure each client workstation with the address of the appropriate Sendmail server. You must also decide how to deal with issues like disproportionate growth or traffic.

To resolve these problems, you can configure one Sendmail service to manage all the other Sendmail servers. You can configure a super service using the 3DNS Controller and all your individual Sendmail servers. This service load balances traffic across all of your Sendmail servers and sends connections to the fastest-performing server at any given time. This allows you, as the administrator, to configure all client workstations to use the same domain name.

```
wideip {
  address 192.168.102.50
  service "smtp"
  name "mx.wip.domain.com"
  pool_lbmode     ratio
  pool {
    name "pool_1"
    type vsb
    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"
    type vsh
    ratio 1
    preferred ratio
    address 192.168.104.50 ratio 2
    address 192.168.105.50 ratio 1
  }
}
```

Figure 5.26 *Configuring one Sendmail service to manage all other Sendmail servers*

Example B

The following example uses multiple resource pools to determine how to distribute connections among three data centers: New York, Los Angeles, and Tokyo. The administrator wants resolution requests that are made to the 3DNS Controller in New York to be resolved to virtual servers in the data center in New York. If the data center in New York fills up and becomes unavailable, the administrator wants to send those resolution requests to virtual servers in the data center in Los Angeles. If both the New York and Los Angeles data centers are full, the administrator wants to send resolution requests to virtual servers in the data center in Tokyo.

Note the use of the null and return to DNS load balancing modes. If requests cannot be resolved using the least connections load balancing mode, those requests are ultimately returned 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"
    type vsb
    ratio 2
    preferred leastconn
    alternate null
    fallback null
    address 192.168.101.50 ratio 2
    address 192.168.101.60 ratio 1
    address 192.168.101.70 ratio 1
  }
```

Figure 5.27 *Distributing connections among three data centers (continued on next page)*

```
  pool {
    name "Los Angeles"
    type vsb
    ratio 1
    preferred leastconn
    alternate null
    fallback null
    address 192.168.102.50 ratio 3
    address 192.168.102.60 ratio 2
    address 192.168.102.70 ratio 1
  }

  pool {
    name "Tokyo"
    type vsb
    ratio 1
    preferred leastconn
    alternate null
    fallback return_to_dns
    address 192.168.103.50 ratio 3
    address 192.168.103.60 ratio 2
    address 192.168.103.70 ratio 1
  }
}
```

Figure 5.28 *Distributing connections among three data centers (continued from previous page)*

The `pool_lbmode` set at the top of the `wideip` statement determines how the connection requests are balanced among the three resource pools (New York, Los Angeles, and Tokyo). The 3DNS Controller first tries to resolve 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 all three modes fail, the 3DNS Controller returns the request to DNS.

Note that in the resource pools above, the `alternate` and `fallback` load balancing methods are set to `null`. Specifying null mode prevents the 3DNS Controller from attempting to do load balancing for the given method. Instead, the 3DNS Controller either goes to the next load balancing method or, if it has cycled through all three load balancing methods for the pool, it then goes to the next resource pool. In this case, because the preferred load balancing method `leastconn` depends on the same metrics data as any static method for `vsb` virtual servers, it is more efficient to perform one load balancing attempt per pool, rather than trying three load balancing attempts before moving to the next available pool.

Also note that the `fallback` load balancing method in the Tokyo pool is set to `return_to_dns`, instead of being set to `null`. Because the `wideip` statement is set to use global availability for load balancing the pools, the 3DNS Controller always utilizes the Tokyo pool last, if at all. If the Tokyo pool fails, the 3DNS Controller returns the resolution request to DNS. This would happen regardless of how the `fallback` method is set in the Tokyo pool, but it is more efficient to set this last fallback to specifically use `return_to_dns`.

Configuring for e-commerce

In this example, the administrator is 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. The administrator 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"
    type vsb
    ratio 2
    preferred qos
    alternate ratio
    address 192.168.101.70 ratio 7
    address 192.168.102.60 ratio 2
  }

  pool {
    name "host_pool"
    type vsh
    ratio 1
    preferred ratio
    address 192.168.104.50 ratio 2
    address 192.168.105.60 ratio 1
  }
}
```

Figure 5.29 *Configuring for e-commerce*

6

3DNS

Web Administration

- **Starting 3DNS administration**
- **Statistics**
- **Administration**

Starting 3DNS administration

The 3DNS Controller comes with a Web Administration tool. This tool gives you a snapshot of your 3DNS Controller network at any given time. With this tool, you can view current information about your network's BIG/ip Controllers, other host machines, virtual servers, paths, and wide IPs. This tool is primarily designed to assist in troubleshooting.

You can start 3DNS Controller administration in either of the following ways:

- From the 3DNS Maintenance Menu, select **Start 3DNS Administration**.
- Type the following command from */usr/contrib/bin*:

  ```
  3dns_admin_start
  ```

The Web Administration tool is divided into two areas:

- **Statistics**
  Presents current statistics for your network.
- **Administration**
  Provides a method of viewing and changing your current configuration.

Setting user access privileges for administration and statistics

You can control user access to the statistics and administration areas using the **Change/Add Users for 3DNS Web Administration** command on the 3DNS Maintenance menu. This menu item opens a script that prompts you to define a user name and password, and also prompts to choose which area(s) the user can access. You can specify that the user has access only to the statistics area, or you can specify the user has access to both the statistics and the administration areas.

Statistics

With the 3DNS Controller administration tool, you can immediately view information about BIG/ip Controllers, other host machines, virtual servers, paths, and wide IPs on your network.

The 3DNS Controller installation includes an HTTP server called ***thttpd***, which is used in the Web Administration tool's display of data. It runs transparently and requires no action on your part. However, if you'd like to change the port to something other than the default of 4999, or make other changes to thttpd, see *thttpd* on page D - 17.

BIG/ip Controller statistics

Click **BIG/ip Controller**s to view the following information about each BIG/ip Controller in your network. The administration tool generates a separate table for each BIG/ip Controller. Each table provides the following information:

Item	Description
Data Center	The IP address or name of the BIG/ip Controller. This address links to a page that displays the `bigip` statement associated with the selected BIG/ip Controller.
OK	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 BIG/ip Controller is new to the 3DNS Controller and that the 3DNS Controller has not yet collected metrics from it.
TTL	The remaining time to live (ttl) before the BIG/ip Controller's data needs to be refreshed.
Seq No.	The number of iQuery packets sent between the specified BIG/ip Controller and the 3DNS Controller.
Packets Out	The total number of IP packets sent by the specified BIG/ip Controller.

Item	Description
Packets In	The total number of IP packets received by the specified BIG/ip Controller.
Packet Rate	The number of packets per second in and out of the BIG/ip Controller during the last sample period.
VS Count	The number of virtual servers managed by the specified BIG/ip Controller.
VS Picks	The number of times a virtual server managed by the BIG/ip Controller received a resolution request from the 3DNS Controller.
Refreshes	The number of times this data was refreshed using the iQuery protocol.
Uptime	The number of days, hours, minutes, and seconds that the specified BIG/ip Controller has been active.
Last Reply	The date and time of the last contact with the specified BIG/ip Controller.

Host statistics

Click **Hosts** to view the following information about the generic host machines in your network. The administration tool generates a separate row for each host machine. The host machine's IP address appears in the third column of each row; the rest of the row provides the following information for that host machine:

Item	Description
OK	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	The remaining time to live (ttl) before a host's metrics data needs to be refreshed.
Interface Address	The IP address associated with the interface that accepts incoming connections for the host. This address links to a page that displays the `host` statement associated with the selected host.
Probe Port	The port that the 3DNS Controller uses to verify whether the virtual server is available.
VS Count	The number of virtual servers managed by the specified host machine.
Prober	The IP address of the machine owning the currently running BIG/3d process.
Protocol	The protocol used for this connection.
Picks	The number of times this host machine was chosen by a wide IP for load balancing.
Refreshes	The number of times this data was refreshed.
Last Refresh	The last time the 3DNS Controller received data about the specified host.

Virtual server statistics

Click **Virtual Servers** to view the following information about each configured virtual server on your network. The administration tool generates a separate row for each virtual server:

Item	Description
OK	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***; yellow indicates that it is ***unavailable***; blue indicates that the virtual is new to the 3DNS Controller and that the 3DNS Controller has not yet collected metrics from it. See *Virtual server decision criteria*, next.
TTL	The remaining time to live (ttl) before a virtual server's metrics data needs to be refreshed.
Type	Whether the specified virtual server is managed by a BIG/ip Controller (VSb) or other host machine (VSh).
Virtual Address	The IP address of the specified virtual server.
Virtual Port	The port number of the specified virtual server.
Ratio	The weighting value for the specified virtual server.
Connections	The number of current connections to the specified virtual server.
Conn Limit	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	The number of nodes currently servicing the specified virtual server.
Enabled	Whether the specified virtual server is available.

Item	Description
Picks	The number of times this virtual server was chosen by a wide IP for load balancing.
Refreshes	The number of times this data was refreshed.
Last Refresh	The last time the 3DNS Controller received data about the specified virtual server.

Virtual server decision criteria

A virtual server is available to be used in a load balancing decision if the following conditions are met:

- The BIG/ip Controller or host machine that governs 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 TTL is specified with the globals sub-statement `vs_ttl`).

Path statistics

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 administration tool generates a separate row for each BIG/ip Controller-to-local DNS path. The total number of paths is shown at the bottom of the table.

Item	Description
TTL	The remaining time to live (ttl) before a path's metrics data needs to be refreshed.
Local DNS	The IP address of the local DNS associated with this path.
BIG/ip	The IP address of the BIG/ip Controller associated with this path.

Item	Description
RTT	The average round trip time (in microseconds) for transactions between the specified BIG/ip Controller and local DNS.
Delta RTT	The difference (in microseconds) between the current known round trip time and the average round trip time.
Completion Rate	The percentage of completed packets versus lost packets, multiplied by 100.
Picks	The number of times the specified path was chosen by a wide IP for load balancing.
Accesses	The number of times the specified path was evaluated to be chosen.
Refreshes	The number of data refreshes for each path.

Local DNS statistics

Click **Local DNS** to view the following information about each configured local DNS on your network. The administration tool generates a separate row for each local DNS.

Item	Description
Rank	A measure of how often this local DNS made resolution requests. ***1*** indicates the local DNS that was used most often, and ***2*** indicates the next most popular, and so on.
Local DNS	The IP address of the local DNS.
3DNS Requests	The number of times the 3DNS Controller received a resolution request from this local DNS.

Item	Description
Probe Protocol	The protocol (either TCP or ICMP) used in communicating with the selected local DNS.
Port	The port number used in communicating with the local DNS.
State	Path probing and path discovery state information. The states are: • ***Needs Probe***: The target has never been probed or scanned. • ***Idle***: Target was successfully probed and is waiting for next probe. • ***In Probe***: Target is currently being probed. • ***Needs Discovery***: Target failed a probe, and now needs to be scanned. • ***In Discovery***: Target is currently being scanned. • ***Suspended***: Target failed the scan and is no longer eligible for probing or scanning.

Wide IP statistics

Click **Wide IPs** to view the following information about each configured wide IP on your network. The administration tool generates a separate row for each wide IP.

Item	Description
Domain Name	The domain name for the specified wide IP. This name links to a page that displays the `wideip` statement associated with the selected domain.
TTL	The `ttl` value specified in the `wideip` statement that is passed back to the local DNS with the ***A*** record.
DNS Address	The A record for the specified domain.

Item	Description
Service	The port 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	The weighting value for the virtual servers owned by BIG/ip Controllers.
VSh Ratio	The weighting value for the virtual servers owned by other host machines.
VSb LB Mode	The load balancing mode in use for the pool of virtual servers owned by a BIG/ip Controller.
VSh LB Mode	The load balancing mode in use for the pool of virtual servers owned by a host machine.
VSb Count	The number of virtual servers owned by a BIG/ip Controller which are used to load balance the specified wide IP.
VSh Count	The number of virtual servers owned by a host machine which are used to load balance the specified wide IP.
Preferred	The number of times a resolution request was resolved using the `preferred` load balancing method specified in the `wideip` statement.
Alternate	The number of times a resolution request was resolved using the `alternate` load balancing method specified in the `wideip` statement.
Fallbacks	The number of times a resolution request was resolved using the `fallback` load balancing method specified in the `wideip` statement.
Returned to DNS	The number of name resolution requests that 3DNS Controller could not resolve. These requests are returned to DNS.
Last Resolution	The last time this name was resolved.

Summary statistics

Click **Summary** to view the following information about your network. The administration tool generates a summary table for each aspect of your network.

General

Item	Description
3DNS Version	The version number of the 3DNS Controller in use.
Max Datasize	The maximum amount of memory that is available for the 3DNS Controller to use.
Start Time	The date and time that the system was booted.
Current Time	The current date and time.
Last Reload	The date and time of the last HUP signal. Corresponds to `ndc reload`.
Last Dump	The date and time of the last INT signal. Corresponds to `ndc dumpdb`.
Total Requests	The number of requests made.
Seconds Up	The number of seconds elapsed since the last reboot.
Average Requests Per Second Since Start Time	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	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.

Primary 3DNS

This table is displayed if the current 3DNS Controller is configured as a data collector. Each 3DNS Controller is a data collector until you designate it a data copier with the globals sub-statement `primary_ip` in the *wideip.conf file*.

Item	Description
Sync DB Interval	The value for `sync_db_interval` as specified in the 3DNS machine's *wideip.conf* file.
Last Dump	The date and time of the last time the data collector's data was successfully sent to a dump file.
Dump File	The name of the file to which the data was sent.
Total Dumps	The number of times that the data collector successfully dumped its data to a file.
Total Dump Errors	The number of times that the data collector was unsuccessful in dumping its data to a file.

Secondary 3DNS

This table is displayed if the current 3DNS Controller is configured as a data copier. A 3DNS Controller is a data copier if its *wideip.conf* file contains the globals sub-statement `primary_ip`.

Item	Description
Sync DB Interval	The value for `sync_db_interval` as specified in the 3DNS machine's *wideip.conf* file.
Last Sync	The date and time of the last time the data copier successfully copied the data collector's data (the dump file).

Item	Description
Sync Primary IP	The IP address of the data collector from which this data copier copies data. It is the value for `primary_ip` as specified in the 3DNS machine's *wideip.conf* file.
Total Syncs	The number of times the data copier successfully copied the data collector's dump file.
Total Sync Errors	The number of times the data copier was unsuccessful in copying the data collector's dump file.

BIG/ip

Item	Description
Total Servers	The number of BIG/ip Controllers controlled by the 3DNS Controller.
Unknown	The number of BIG/ip Controllers for which the status is not known.
Up	The number of BIG/ip Controllers controlled by the 3DNS Controller currently marked ***up***.
Down	The number of BIG/ip Controllers controlled by the 3DNS Controller currently marked ***down***.
Waiting	The number of BIG/ip Controllers controlled by the 3DNS Controller currently in waiting mode.
Alert	The number of BIG/ip Controllers controlled by the 3DNS Controller currently in alert mode.
Panic	The number of BIG/ip Controllers controlled by the 3DNS Controller currently in panic mode.
Average Packet Rate	The average number of packets per second in and out of the BIG/ip Controller.
Average Connections	The average number of connections from the start time to the current time.
Average Nodes	The number of total nodes up divided by the number of BIG/ip Controllers.

Host

Item	Description
Total Hosts	The number of other host machines controlled by the 3DNS Controller.
Up	The number of other host machines controlled by the 3DNS Controller currently marked ***up***.
Down	The number of other host machines controlled by the 3DNS Controller currently marked ***down***.

Virtual Servers

Item	Description
Total Virtual Servers	The total number of virtual servers.
Total BIG/ip Virtual Servers	The number of virtual servers managed by BIG/ip Controllers.
--Up	The number of BIG/ip virtual servers that are up.
--Down	The number of BIG/ip virtual servers that are down.
Total Host Virtual Servers	The number of virtual servers managed by a host machine.
--Up	The number of host virtual servers that are up.
--Down	The number of host virtual servers that are down.

Wide IP

Item	Description
Total Wide IPs	The number of defined wide IPs.
Total Requests	The number of name resolution requests sent to the 3DNS Controller.
Total Non-Wide IP Requests	The number of regularDNS requestnot intended to be load balanced.
Total Wide IP Requests	The number of requests sent to a wide IP for resolution and load balancing.
Total Resolved	The number of successful name resolutions.
--By Preferred	The number of resolutions made using the preferred load balancing method.
--By Alternate	The number of resolutions made using the alternate load balancing method.
--By Fallback	The number of resolutions made using the fallback load balancing method.
Total Returned to DNS	The number of name resolution requests that are returned to DNS.

Local DNS

Item	Description
Total Local DNS	The number of local DNS systems accessed by the 3DNS Controller.
Probed by ICMP	The number of local DNS systems accessed by the 3DNS Controller that are probed by ICMP.
Probed by TCP	The number of local DNS systems accessed by the 3DNS Controller that are probed by TCP.
Probed by UDP	The number of local DNS systems accessed by the 3DNS Controller that are probed by UDP. Not implemented for this release.
--Needs Probe	The number of local DNS systems that have not been probed.
--Idle	The number of local DNS systems that were successfully probed and are waiting for the next probe.
--In Probe	The number of local DNS systems that are currently being probed.
--Needs Discovery	The number of local DNS systems that failed a probe.
--In Discovery	The number of local DNS systems that are currently being scanned.
--Suspended	The number of local DNS systems that failed the scan and are no longer eligible for probing or scanning.
Ports Discovered	The number of local DNS systems whose ports have been discovered.

Path

Item	Description
Total Paths	The number of paths used by the 3DNS Controller.
Paths Probed Successfully	The number of paths that were successfully probed.
Fresh Paths	The number of new paths.
Current Average RTT	The average of current RTT metrics for all paths.
Overall Average RTT	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	The average of current metrics for the percentage of completed packets versus lost packets.
Overall Average Completion Rate	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.
Total Picks	The number of times (for all paths) where the path's data resulted in the corresponding BIG/ip Controller's virtual server being chosen for a connection.
Total Accesses	The number of times all paths were considered when performing dynamic load balancing.
Average Outstanding Requests	The number of iQuery requests made by the 3DNS Controller to a particular BIG/ip Controller that were dropped or not serviced within the `timer_get_data` timeframe.

Global variable statistics

Click **Globals** to view information about the current and default values for each `globals` sub-statement, and to find out if any changes you made require that you restart *named*.

Documentation

For more information on 3DNS Controllers and utilities, click **Man Pages**, **Users Guide,** or **Release Notes**. Note that opening the online Users Guide takes a few moments. The file is rather large, and the viewing software (Adobe Reader) must be started.

Administration

From the Administration area, you can view and edit the *wideip.conf* file, change global variable settings, update the current statistics and configuration settings in the *wideip.conf* file, start and stop metrics collection on paths, and reset all statistics.

Commands

You can perform the following tasks using the buttons under **Commands** in the left frame:

Item	Description
Reset Statistics	Sets all statistics values to zero and begins collecting new statistical data.
Start Metrics Collection	Activates the process of collecting statistical data.
Stop Metrics Collection	Deactivates the process of collecting statistical data.

Configuration

In addition to viewing collected information about your network, you can use the 3DNS Controller administration tool to view and change your configuration file.

View wideip.conf

Displays the contents of your *wideip.conf* file.

Edit wideip.conf

Opens the *wideip.conf* file in an edit window. Once you are finished making changes, click **Update**.

There are three limitations to editing your *wideip.conf* file with this tool:

- You cannot edit a *wideip.conf* file that is larger than 64 Kb.
- You can only change or add items in a *wideip.conf* file; you cannot remove items from the file.
- If you use incorrect syntax in the *wideip.conf* file, the file fails parsing, and the erroneous text is displayed for review.

Edit Globals

Lets you view and edit individual variables in your `globals` statement without loading the *wideip.conf* file.

To edit global variables in this window

1. Click the variable name.
2. Make your edits.
3. Click **Update**.

The **Change Requires Restart** column indicates whether you must restart *named* for changes to take effect.

Update Database

Creates two files:

- */var/3dns/etc/wideip.conf.dynamic*
  This file stores the wide IP definitions, path data, and local DNS data.

- */var/3dns/etc/wideip.conf.static*
  This file contains only the globals, bigip statements, hosts statements, and wideip statements.

For information on dynamic and static *wideip.conf* files, see *Working with static and dynamic wideip.conf files*, on page C-2.

Restart

Restarts the *named* process. This is equivalent to issuing the `ndc restart` command.

If you change your configuration file, click **Restart** for the changes to take effect. When the 3DNS Controller restarts, it re-reads the configuration information.

7

3DNS

Statements and Comments

- Statements
- Comments

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 Appendix C, *The wideip.conf File.*

The 3DNS platform supports the following top-level statements:

- `globals`
  Controls global 3DNS Controller configuration options and sets defaults for other statements. This statement may be used only once per configuration.
- `bigip`
  Defines a BIG/ip® Server Array Controller managed by the 3DNS Controller.
- `host`
  Defines a single network server or other server array controller.
- `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 new 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**
  The `globals` statement should appear first in the *wideip.conf* file, followed by `bigip` and `host` statements. The `wideip` statements should appear next, following by the `topology` statement.

- **Address specification**
  Wherever an address is required in a statement, the address must precede all other possible sub-statements.
- **Port specification**
  Wherever a port specification is required in a statement, it must immediately follow the address specification. The exception is the host statement, where the port specification follows the `probe_protocol` sub-statement. In all other cases, the port specification 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.

- **Pool specification**
  A pool is a set of virtual servers defined and owned by a BIG/ip Controller or other host machine. Acceptable values are `vsb` (virtual servers owned by a BIG/ip Controller) and `vsh` (virtual servers owned by a host machine). The default is `vsb`. You can have both types of virtual servers in the same `vsh` pool definition, but you can only include virtual servers owned by a BIG/ip Controller in a `vsb` pool. Note that `vsh` pools can only use static load balancing modes.
- **`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 C-16.

The globals statement

The globals statement sets up global options to be used by the 3DNS Controller, and must appear before any bigip, host, or wideip statements in the *wideip.conf* file. Each globals sub-statement has a default setting. You do not need to edit the globals statement unless you want to change a sub-statement's 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.

The globals statement should appear only once in a configuration file; if the 3DNS Controller finds more than one occurrence, the 3DNS Controller generates an error, alerting you that your configuration contains multiple globals statements.

However, if you use a globals sub-statement more than once within the globals block, the 3DNS Controller uses the last listed value and does not generate an error. For example, if your globals block contains the following lines, the 3DNS Controller uses the value 50:

```
globals {
  host_ttl 100
  host_ttl 50
}
```

Syntax for globals statement

The globals statement supports the following sub-statements. When you define a globals statement, you need to include only those sub-statements that you want to change from the default.

```
globals {
  [ primary_ip <ip_addr> ]
  [ sync_db_interval <number> ]
  [ check_static_depends < yes | no> ]
  [ timer_get_bigip_data <number> ]
  [ timer_get_host_data <number> ]
  [ timer_get_vs_data <number> ]
  [ timer_get_path_data <number> ]
  [ bigip_ttl <number> ]
  [ host_ttl <number> ]
  [ vs_ttl <number> ]
  [ path_ttl <number> ]
  [ rtt_timeout <number> ]
  [ rtt_sample_count <number> ]
  [ rtt_packet_length <number> ]
  [ rtt_probe_protocol < icmp | tcp > ]
  [ rx_buf_size <number> ]
  [ tx_buf_size <number> ]
  [ timer_check_keep_alive <number> ]
  [ qos_coeff_rtt <number> ]
  [ qos_coeff_completion_rate <number> ]
  [ qos_coeff_packet_rate <number> ]
  [ qos_coeff_topology <number> ]
  [ default_alternate < rr | ratio | ga | random | return_to_dns
    | topology |  null > ]
  [ default_fallback < rr | ratio | ga | random | return_to_dns
    | topology |  null > ]
```

Figure 7.1 *Syntax for globals statement (continued on next page)*

```
  [ fb_respect_depends < yes | no > ]
  [ fb_respect_acl < yes | no > ]
  [ encryption < yes | no > ]
  [ encryption_key_file <string> ]
  [ path_hi_water <number> ]
  [ path_lo_water <number> ]
  [ path_duration <number> ]
  [ path_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_discovery_method < short | wks | full | all > ]
  [ path_max_refreshes <number> ]
}
```

Figure 7.2 *Syntax for globals statement (continued from previous page)*

Example

```
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 to tcp probing if icmp
                             // fails
  multiplex_iq yes           // Source port is the same as
                             // destination port for iQuery
  use_alternate_iq_port yes  // Use IANA registered port for
                             // iQuery
}
```

Figure 7.3 *Example syntax for globals statement*

Definition of globals sub-statements

Each globals sub-statement supports the parameters described below.

Primary IP address

You include this sub-statement only when configuring a 3DNS Controller as a data copier.

Parameter	Description	Default
primary_ip	Specifies the IP address of the data collector from which the current data copier retrieves metrics information.	0

Synchronization

The synchronization sub-statement specifies how the current 3DNS Controller handles synchronizing its database with the other 3DNS Controllers in the network.

Parameter	Description	Default
`sync_db_interval`	On a data collector, specifies the amount of time (in seconds) between updates to the database. On a data copier, specifies how often to copy and read the data collector's database file. You can enter a value between 60 and 4294967295.	`600`

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. This check is performed only when the 3DNS Controller uses a static load balancing mode. If the 3DNS Controller is using a dynamic load balancing mode, an availability check is always performed.

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.	`yes`

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_bigip_data`	The 3DNS Controller refreshes the BIG/ip Controller information at intervals determined by `timer_get_bigip_data`. You can enter a value between 0 and 4294967295 seconds.	`20`
`timer_get_host_data`	The 3DNS Controller refreshes other host machine information at intervals determined by `timer_get_host_data`. You can enter a value between 0 and 4294967295 seconds.	`90`
`timer_get_vs_data`	The 3DNS Controller refreshes virtual server information at intervals determined by `timer_get_vs_data`. You can enter a value between 0 and 4294967295 seconds.	`30`
`timer_get_path_data`	The 3DNS Controller refreshes path information (for example, round trip time or ping packet completion rate) at intervals determined by `timer_get_path_data`. You can enter a value between 0 and 4294967295 seconds.	`120`
`timer_check_keep_alive`	The 3DNS Controller queries remote BIG/ip Controllers every `timer_check_keep_alive` seconds. You can enter a value between 0 and 4294967295.	`60`

Data timeouts

These sub-statements set the amount of time for which metrics information is considered valid. Once a timeout is reached, the 3DNS Controller refreshes the information. Note that on a data copier, it is important that you set all TTL values to be greater than the value set for `sync_db_interval`.

Parameter	Description	Default
`bigip_ttl`	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`	The amount of time (in seconds) that other 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`	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`	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`

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** listener 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 or TCP protocol.	`icmp`

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 4294967295.	`16384`
`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 4294967295.	`8192`

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 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)]
```

The coefficients for the score computation are defined as globals, but may be overridden within a `wideip` statement.

Parameter	Description	Default
`qos_coeff_rtt`	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`	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`	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`	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`

Load balancing

Parameter	Description	Default
default_ alternate	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 fall-back mode.	no
fb_respect_acl	Determines whether the 3DNS Controller imposes access control when load balancing switches to the specified fall-back mode.	no

For more information on selecting a load balancing mode, see Chapter 5.

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

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`	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. This sub-statement can be overridden within the `host` statement.	`0.0.0.0`

◆ WARNING

We recommend that you define a default prober if the 3DNS Controller manages virtual servers on hosts. If you do not define a default prober, and you do not define probers for all hosts, you may 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`	The UDP receive buffer size. The value is overridden only if it is larger than the one first assigned by the kernel.	`8192`
`resolver_tx_buf_size`	The TCP send buffer size.	`16384`

Reaping

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` *and* `path_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 for reaping.	`varies`
`path_lo_water`	Specifies the low water mark for reaping.	`varies`
`path_duration`	An event is triggered every `path_duration` seconds that calls the reaping function. You can enter a value between 3600 and 2419200 seconds.	`345600`
`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`

iQuery port options

Parameter	Description	Default
`use_alternate_iq_port`	Determines whether the 3DNS Controller runs iQuery traffic on port 245 (the port used in older configurations), or on the new 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** utility. To force iQuery traffic to use port 4353 for all incoming iQuery traffic, change this setting to `yes`.	`no`

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.	`no`
`check_dynamic_depends`	Specifies that the 3DNS Controller checks the availability of a path before it uses the path for load balancing.	`yes`
`rtt_probe_dynamic`	Determines whether the 3DNS Controller attempts a second probe using the alternate probe protocol if the probe protocol specified by `rtt_probe_protocol` fails during the first probe.	`no`
`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_discovery_method`	Determines which ports to scan.	`short`
`path_max_refreshes`	Determines the maximum number of times the 3DNS Controller requests new data for the path.	`0` (no limit)

The bigip statement

The `bigip` statement defines the characteristics associated with a particular BIG/ip Controller.

A `bigip` statement contains the following information:

- The IP address of the BIG/ip Controller.
- The set of virtual servers that are available on the specified 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.

Syntax for bigip statement

The `bigip` statement syntax includes the following sub-statements.

```
bigip {
 address <ip_addr>
 vs {
  address <ip_addr>:<port_number>
  port <port_number> | service <"service_name">
  [ ratio <number> ]
   [ translate {
    address <ip_addr>
    port <port_number>|service <"service_name">
     }
  }
}
```

Figure 7.4 *Syntax for bigip statement*

Example

```
bigip {
  // New York
  address 192.168.101.40
  vs {
    address 192.168.101.50
    port 80
    translate {
      address 10.0.0.50
      port 80
    }
  }
}
```

Figure 7.5 *Example syntax for bigip statement*

Definition of bigip sub-statements

The bigip sub-statements specify information about the virtual servers managed by a BIG/ip Controller.

Parameter	**Description**
`address`	In the context of a `bigip` statement, `address` specifies the IP address of the BIG/ip Controller.
`vs`	Indicates the start of a virtual server definition. Once you define a virtual server here (including specifying the address and port), you can then use this virtual server in a `wideip` definition.

Parameter	Description
address	As part of a virtual server (vs) definition, address specifies the IP address of a virtual server owned by this BIG/ip Controller. Note that the virtual server’s address must be listed first, before port, service, or ratio values.
port or service	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.
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 name and port/service information for the translated IP addresses.

The host statement

The host statement defines information about the host itself, including its IP address, and also defines information about the individual virtual servers associated with it.

Syntax for host statement

```
host {
  address <ip_addr>
  probe_protocol <tcp | icmp>
  port <port>
  [prober <ip_addr>]
  vs {
    address <ip_addr>:<port_number>
    port <port_number> | service <"service_name">
    probe_protocol <tcp | icmp>
  }
}
```

Figure 7.6 *Syntax for host statement*

Example

```
host {
  // Tokyo
  address 192.168.104.40
  vs {
    address 192.168.104.50:80
    probe_protocol  tcp
  }
}
```

Figure 7.7 *Example syntax for host statement*

Definition of host sub-statements

The host sub-statements define information about the virtual servers managed by a host server. The host sub-statements also define the method used to ping the host server to verify if it is available.

Parameter	Description
`address`	In the context of a host statement, `address` specifies the host machine's IP address.
`probe_protocol`	The protocol method to use for probing this host: TCP or ICMP.
`port`	The port used to probe this host if `probe_protocol` is set to TCP.
`prober`	The IP address of the machine probing the host. This IP address points to either a BIG/ip Controller or a 3DNS Controller that runs the **big3d** utility. The **big3d** utility 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 in the host sub-statement, the 3DNS Controller uses the `prober <ip_addr>` parameter defined in the `globals` statement.

Parameter	Description
`vs`	Indicates the start of a virtual server definition. Once you define a virtual server here (including specifying the `address`, `port`, and `probe_protocol` values), you can then use this virtual server in a wide IP definition. Each `host` statement can include multiple virtual servers, but must always include at least one virtual server.
`address`	As part of a virtual server (vs) definition, `address` specifies the IP address of a virtual server owned by this host machine.
`port or service`	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.

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).

A wide IP contains one or more pool sub-statements that define individual load balancing pools. A load balancing pool is a group of virtual servers that the 3DNS Controller load balances, and it is limited only in that the virtual servers included in the pool must be of the same type (either BIG/ip virtual servers or host virtual servers).

A `wideip` statement specifies the following:

- A domain name and a key.
- A set of virtual servers accessing all the instances of a mirrored service.
- Parameters configuring the algorithm which chooses the best virtual server for each transaction.

Syntax for wide IP 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>
    } ]
  [ pool_lbmode <rr | ratio | ga | random> ]
  pool {
    name <"pool_name">
    type <vsb | vsh>
    [ ratio <pool_ratio> ]
    [ preferred <rr | ratio | ga | topology |
      random | leastconn | packet_rate |
      completion_rate | rtt | qos> ]
    [ alternate <rr | ratio | ga | topology |
      random | return_to_dns | null> ]
    [ fallback <rr | ratio | ga | topology |
      random | leastconn | packet_rate |
      completion_rate | rtt | qos> ]
    address <vs_addr>[:<port>] [ratio <weight>]
    }
  }
```

Figure 7.8 *Syntax for wideip statement*

Example

```
//
wideip {
  address 192.168.101.50
  service "http"
  name "www.wip.domain.com"
  qos_coeff {
    rtt              21
    completion_rate   7
    packet_rate       5
    topology          1
   }

  pool {
    name "pool_1"
    type vsb
    ratio 2
    preferred qos
    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 7.9 *Example syntax for wideip statement*

Definition of wideip sub-statements

Wide IP sub-statements defines 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 specifies the wide IP key (see *Understanding the wide IP key*, on page 4-28). They also specify the pool of virtual servers that the wide IP load balances.

Parameter	**Description**
`address`	A key that represents one valid virtual server IP address from the set which services this wide IP. This key is also listed as the A record in the zone file for the domain. See *Understanding the wide IP key*, on page 4-28.
`port or service`	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`	The domain name for this wide IP (for example, `"www.wip.f5.com"`). All names must be enclosed in quotation marks.
`alias`	An alternate name for this wide IP. All names must be enclosed 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`	The amount of time (in seconds) that the specified wide IP's information is used by the 3DNS Controller for name resolution and load balancing.
`port_list`	Specifies a list of ports that must be available before the 3DNS Controller can send connections to the specified address.
`qos_coeff`	Relative weighting for each load balancing method in calculating the Quality of Service mode. Each load balancing mode is described in the next table.
`pool_lbmode`	The load balancing mode to use to balance requests over all pools.
`pool`	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.

Parameter	Description
type	The type of pool: vsb (virtual servers owned by a BIG/ip Controller) and vsh (virtual servers owned by a host machine). The default is vsb. You cannot include both types of virtual servers in the same pool definition.
ratio	As part of a pool definition, ratio specifies the default weighting to use with respect to other pool types.
preferred	The load balancing mode to use for the specified pool. Each acceptable value is described in the next table.
alternate	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_alternate, a globals sub-statement, on page 7-13.
fallback	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 7-13.
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	An optional part of specifying a virtual server. 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.

Load balancing mode

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.

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 load balancing mode defined in the globals statement (see page 7-13).

Parameter	Description
completion_rate	Least packets dropped (or timed out). Valid for vsb pools only, and only in a preferred or fallback sub-statement.
global_availability	First virtual server listed in the wide IP definition. Valid for both vsb and vsh pools.
leastconn	Least number of current connections for a virtual server. Valid for vsb pools only, and only in a preferred or fallback sub-statement.
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 only in an alternate or fallback sub-statement.
packet_rate	Least packets per second the BIG/ip Controller is processing. Valid for vsb pools only, and only in a preferred or fallback sub-statement.
qos	User definable metric that includes a combination of packet rate, completion rate, RTT, and topology. Valid for vsb pools only, and only in a preferred or fallback sub-statement.
random	Virtual server chosen at random from the wide IP set of virtual servers. Valid for both vsb and vsh pools.
ratio	Distributed percentages. Valid for both vsb and vsh pools.
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. Valid only in an alternate or fallback sub-statement.

Parameter	Description
`rr`	Circular and sequential. Valid for both `vsb` and `vsh` pools.
`rtt`	Shortest timed ICMP packet from a virtual server's BIG/ip Controller to the requestor's local DNS. Valid for `vsb` pools only, and only in a `preferred` or `fallback` sub-statement.
`topology`	Distributes connections based on the proximity of a local DNS to a particular data center.

Use the following equation to configure the QOS load balancing mode:

```
A (1/packet rate) + B (1/rtt) + C (completion rate) + D (topology)
```

For more information on each mode and some load balancing examples, see Chapter 5, *Load Balancing*.

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 *Topology-based access control*, on page 5-15.

Syntax for topology statement

```
topology {
  acl_threshold <1 | 0>
  limit_probes <yes |no>
  longest_match <yes | no>
    <server cidr> <LDNS cidr> <score>
}
```

Figure 7.10 *Syntax for topology statement*

Definition of topology sub-statements

Parameter	**Description**
`acl_threshold`	Provides a hook for administrators to set up access control to data centers based on local DNS IP addresses by specifying a score threshold. Any server/local DNS matching a list record with a score below this threshold is interpreted as if the virtual server were unavailable.
`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 `acl_threshold` value and the topology map scores.
`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.
`mask virtual server`	The server mask for a given data center. This is one of two values used to determine the longest match.
`mask LDNS`	The local DNS mask. This is one of two values used to determine the longest match.
`mask score`	The mask score, which is used for the comparison of virtual servers when the topology load balancing mode is employed.

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 7.11** 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 7.12 *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 7.13 *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 7.14 *Syntax for shell style comments*

8

Additional System and Network Configuration

- **Changing passwords for the 3DNS Controller**
- **Configuring Sendmail**
- **Enabling dynamic routing**

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.

Changing the 3DNS Controller password

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 Return.
3. To confirm the password, retype it and press Return.

Changing passwords and adding new user IDs

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:

1. Start the 3DNS menu by entering the following command from */usr/contrib/bin*:

   ```
   3dnsmaint
   ```

2. From the 3DNS Maintenance Menu, select **Add 3DNS Administration Password**.

This starts the *3dns_web_passwd* script, which lets you provide access to the 3DNS Web Administration site for selected users only, and assigns passwords for those users. If you don’t use this script, all users have access to the Web Administration site.

Configuring Sendmail

You can configure the 3DNS Controller to allow electronic mail to be sent from the system. This configuration must be completed if the 3DNS Controller is to send electronic mail to the administration workstation or to an alphanumeric pager. The 3DNS platform includes an example configuration file that is suitable for most sites. Before you use this configuration file, however, you do have to customize it for your network environment.

Customizing the */etc/sendmail* file

When you customize this file, you enter the name of the mail relay server.

Finding the mail relay in your network

1. From a machine capable of name resolution, type the following on the command line:

   ```
   3dns: /etc# nslookup
   ```

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

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

3. Next, query for the mail relay server for your domain using the following command:

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

The information returned includes the name of the mail exchanger.

Setting up Sendmail

1. Copy */etc/sendmail.cf.off* to */etc/sendmail.cf*.
2. Edit */etc/sendmail.cf* and set the DS variable to the name of the mail exchange server.
3. 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
```

Including this line in the */etc/crontab* file sets Sendmail to flush the outgoing message queue for any email that could not be delivered immediately. Because the 3DNS Controller does not accept email from external sources, there is no need to run the *Sendmail* daemon. Queue flushes are issued via *crontab.*

4. Save and close the */etc/crontab* file.
5. Open the */etc/aliases* file.
6. In the */etc/aliases* file, create an entry for root to point to an administrator at your site. For example:

   ```
   root: networkadmin@domain.com
   ```

   Because the 3DNS Controller does not accept local email, bounces or undelivered messages go unnoticed. This requires that the administrator is notified when a message is bounced or undelivered.
7. Save and close the */etc/aliases* file.
8. Run the **`newaliases`** command to generate the new aliases database using the information you just added.
9. Reboot the 3DNS Controller.

Enabling dynamic routing

The 3DNS platform includes the GateD daemon, which is disabled by default. To enable the 3DNS Controller to accept dynamic routing updates from your routers, you must first create the appropriate configuration file, */etc/gated.conf*.

Enabling the GateD daemon

You enable the GateD daemon on the 3DNS Controller by typing the following at the command line prompt:

```
3dns# gated
```

Editing the */etc/netstart* file

Next, you need to edit the */etc/netstart* file and change the definition of the gated variable as shown below:

```
gated=YES
```

The 3DNS Controller is now configured to accept dynamic route updates from your router.

◆ **Note**

Certain network environments may require that you modify the routing tables or your router. If you have communication problems between your router and the 3DNS Controller, please contact Technical Support at F5 Networks, Inc.

A

Glossary

Term	Definition
BIG/ip Controller	A Service Array Controller that monitors each server for application availability and performance, and automatically routes incoming queries to the most available server.
big3d	The listener that runs on each BIG/ip Controller and responds to 3DNS Controller queries.
BIND (Berkley Internet Name Domain)	The most common implementation of DNS which provides a system for matching domain names to IP addresses.
daemon	A program that runs in the background on UNIX systems and responds to requests from services or from other hosts on a network.
data collector	Any 3DNS Controller that collects data. Each 3DNS Controller is a data collector until you designate it a data copier with the globals sub-statement `primary_ip`.
data copier	A 3DNS Controller that copies data from data collectors at intervals specified with the globals sub-statement `sync_db_interval`. Any 3DNS Controller that contains the globals sub-statement `primary_ip` is a data copier.
DNS (Domain Name System)	A distributed database that maps IP addresses to host names.
DNS server	See *name server.*
encryption key	The sequence of data that prevents unauthorized access to other data.
fallback address	See *wide IP key.*
FDDI (Fiber Distributed Data Interface)	A multi-mode protocol for transmitting data on optical-fiber cables up to 100Mbps.
F-Secure SSH	An encryption utility that allows secure shell (SSH) connections to a remote system such as the BIG/ip Controller.
gateway	Hardware and/or software that forwards data between two networks.
host	Any computer on a network that makes services available to other computers on the network.

Term	Definition
host machine	For the purposes of this manual, "host machine" refers to a single network server or a server array controller other than a BIG/ip Controller.
HUP	A BIND name server signal. It causes the name server to reload configuration files. Use this signal after modifying the name server's boot file or one of its database files for the changes to take effect. You can also send this signal to BIND 4.93 secondary name servers to update its secondary zones.
ICMP (Internet Control Message Protocol)	An Internet communications protocol. This protocol provides information relevant to IP packet processing and error correction.
INT	A BIND name server signal. It saves a copy of the name server's database to a file called *named_dump.db*. This file is located in */var/tmp* or */usr/tmp*, depending on your configuration.
InterNIC	A US organization that registers domain names and IP addresses and distributes information about the Internet. The InterNIC Internet address is *rs.internic.net*.
iQuery	A UDP-based protocol used to communicate and exchange information between BIG/ip Controllers and 3DNS Controllers.
local DNS	A DNS server making the name resolution request on behalf of a client. From the perspective of the 3DNS Controller, the local DNS is the source of the name resolution request.
name server	A computer that can answer DNS queries. Name servers contain information about some part of the DNS, and they make that information available to clients. Also called DNS server.
named (name server daemon)	The name server daemon, which manages domain name server software.
node	A specific server in the array managed by a BIG/ip Controller.
path	A logical route between a BIG/ip Controller and a local DNS.

Term	Definition
pool	A group of virtual servers defined and owned by BIG/ip Controllers and other host machines that are load balanced as part of a wide IP.
primary DNS	The name server that manages the authoritative domain name information for a zone.
QOS (Quality of Service)	A dynamic load balancing mode that bases connection distribution on a configurable combination of the packet rate, completion rate, round trip time, and topology modes.
resolution	In DNS terminology, the process by which a name server retrieves data that is requested by a resolver, and sends it to the resolver.
resolvers	In DNS terminology, the clients that accesses name servers. A resolver queries a name server, interprets the responses, and returns the information to the program that requested it.
resource record	The building blocks of the DNS. 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
RTT (Round Trip Time)	A calculation of the time (in microseconds) that the local DNS takes to respond to a probe issued by the big3d utility.
secondary DNS	A name server that gets DNS data from the name server that is authoritative for the DNS zone.
TTL (Time to Live)	A variable that controls how long information is kept in the cache and used in making decisions.
virtual address	An IP address associated with one or more virtual servers managed by the BIG/ip Controller.
virtual port	One component of a virtual server. The virtual port number should be the same TCP or UDP port number that is known to client programs.
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 host machine.
wide IP	Manages and balances information on BIG/ip Controllers or other host machines by mapping a domain name to a load balancing method and a set of virtual servers.

Term	Definition
wide IP key	The wide IP key is sometimes referred to as the fallback address. The wide IP key is the same address as the domain name address (the DNS ***A*** record) and the wide IP address.
WKS (Well-Known Services)	A type of resource record that describes the services usually provided by a particular protocol on a particular port.
zone files	A DNS term. A database file that stores 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).

B

3DNS Controller Configuration Checklist

Overview

This appendix provides a configuration checklist, which you should complete before you begin to install a 3DNS Controller.

You may want to make photocopies of the checklist, and use one form for each 3DNS Controller in your network. Keep the completed checklists for future reference.

3DNS Controller Configuration Checklist

Interface IP Address

Domain name (for example, *test.net*)

IP address or name of primary DNS

Create and delegate new subdomain (for example, *wip.test.net*) on the primary DNS for use by the 3DNS Controller. List subdomains here:

Checklist, continued

Identify domains to be load balanced (for example, *www.test.net* and *ftp.test.net*):

Virtual servers managed by BIG/ip Controllers to be assigned to wide IPs:

Virtual servers managed by other host machines to be assigned to wide IPs:

C

3DNS

The wideip.conf File

Overview

The 3DNS Controller configuration file is called */etc/wideip.conf*. It consists of two types of information: statements and comments.

You must edit the 3DNS Controller configuration file to suit your network. Use the sample configuration file */etc/wideip.conf.samp*, which is included later in this chapter, as a guide. The */etc/wideip.conf* file describes the BIG/ip Controllers, other host machines, and wide IPs that are managed by the 3DNS Controllers.

At the minimum, your *wideip.conf* file must contain the following:

- At least one virtual server, which can be defined in either a `bigip` or `host` statement
- A `wideip` statement

Refer to Chapter 7, *Statements and Comments*, for information on valid statements and sub-statements, as well as for the proper syntax.

Working with static and dynamic wideip.conf files

You have the option of maintaining your original *wideip.conf* file separately from a dynamic *wideip.conf* file that includes the most recent path and local DNS information.

The 3DNS Maintenance menu includes two commands to support this feature: **Use Dynamic wideip.conf**, and **Use Static wideip.conf**:

- **Use Dynamic wideip.conf**
  Renames the existing */etc/wideip.conf* file to */var/3dns/etc/wideip.conf.ORIG* if it is found to be in the Initial state, and it also creates a link from */etc/wideip.conf* to */var/3dns/etc/wideip.conf.dynamic*.

- **Use Static wideip.conf**
  Renames the existing */etc/wideip.conf* file to */var/3dns/etc/wideip.conf.ORIG* if it is found to be in the Initial state, and it also creates a link from */etc/wideip.conf* to */var/3dns/etc/wideip.conf.static*.

You can manually edit the */etc/wideip.conf* file in a text editor and the correct file is modified in preparation for a restart.

◆ **Note**

You must restart the system before implementing any other dynamic commands to avoid losing changes to the edited ***wideip.conf****. To avoid any possible loss of any changes, use the* ***Edit 3DNS Configuration*** *command from the menu or the* ***edit_wideip*** *script.*

To open the */etc/wideip.conf* file

1. From the command prompt, change to the */etc* directory by typing:

   ```
   cd /etc
   ```

2. Use a text editor such as vi or pico to open the *wideip.conf* file. For example, if you use vi, type the following:

   ```
   vi wideip.conf
   ```

Example: 3DNS Controller configuration file

The following is an example of a 3DNS Controller configuration file. Note that very few global parameters are listed. You do not need to include each global parameter; you should include only those parameters for which you want to specify a value other than the default.

Note that this sample file contains examples of common configurations and each load balancing mode. Each load balancing example is further described in *Example syntax for global availability*, starting on page 5-30.

```
#
# Sample /etc/wideip.conf
#
# Related files are:
# /etc/named.conf
# /var/namedb/db.wip.domain.com
#

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 to tcp probing if icmp fails
  multiplex_iq yes           // Source port is the same as
                             // destination port for iQuery
  use_alternate_iq_port yes  // Use IANA registered port for iQuery
}

// The New York BIG/ip is behind a firewall and the virtual servers
// need to be translated

bigip {
  // New York
  address 192.168.101.40
  vs {
    address 192.168.101.50
    port 80
    translate {
      address 10.0.0.50
      port 80
    }
  }

  vs {
```

```
  address 192.168.101.50
  port 25
  translate {
    address 10.0.0.50
    port 25
  }
}

vs {
  address 192.168.101.60
  port 80
  translate {
    address 10.0.0.60
    port 80
  }
}

vs {
  address 192.168.101.60
  port 21
  translate {
    address 10.0.0.60
    port 21
  }
}

vs {
  address 192.168.101.70
  port 80
  translate {
    address 10.0.0.70
    port 80
  }
}

vs {
  address 192.168.101.70
```

```
    port 443
    translate {
      address 10.0.0.70
      port 443
    }
  }
}


bigip {
  // Los Angeles
  address 192.168.102.40
  vs {
    address 192.168.102.50:80
  }

  vs {
    address 192.168.102.50:25
  }

  vs {
    address 192.168.102.60:80
  }

  vs {
    address 192.168.102.60:443
  }

  vs {
    address 192.168.102.60:21
  }

  vs {
    address 192.168.102.70:80
  }
}
```

```
bigip {
  // Tokyo
  address 192.168.103.40
  vs {
    address 192.168.103.50:80
  }

  vs {
    address 192.168.103.50:25
  }

  vs {
    address 192.168.103.60:80
  }

  vs {
    address 192.168.103.60:21
  }

  vs {
    address 192.168.103.70:80
  }
}


host {
  // Tokyo
  address 192.168.104.40
  vs {
    address 192.168.104.50:80
    probe_protocol  tcp
  }

  vs {
    address 192.168.104.50:443
    probe_protocol  tcp
  }

  vs {
    address 192.168.104.50:25
    probe_protocol  tcp
  }
}


host {
  // New York
  address 192.168.105.40
  port 80
  probe_protocol tcp
  prober 192.168.103.40           // Use the prober in Tokyo

  vs {
    address 192.168.105.50:80
    probe_protocol  tcp
  }

  vs {
    address 192.168.105.50:25
    probe_protocol  tcp
  }

  vs {
    address 192.168.105.60:80
    probe_protocol  icmp
  }

  vs {
    address 192.168.105.60:443
    probe_protocol  icmp
  }
}
```

```
//
wideip {
  address 192.168.101.50
  service "http"
  name "www.wip.domain.com"
  qos_coeff {
    rtt              21
    completion_rate   7
    packet_rate       5
    topology          1
   }

  pool {
    name "pool_1"
    type vsb
    ratio 2
    preferred qos
    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
  }
}


// Global availability
wideip {
  address 192.168.101.60
  port 80
  name "cgi.wip.domain.com"
  pool {
    name "mypool"
    type vsb
    preferred ga
    address 192.168.101.60 // New York
    address 192.168.102.60 // Los Angeles
    address 192.168.103.60 // Tokyo
  }
}


// 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"
    type vsb
    preferred rtt
    alternate topology
    address 192.168.101.60 // New York
    address 192.168.102.60 // Los Angeles
    address 192.168.103.60 // Tokyo
  }
}


// Least connections with ratio as an alternate
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 // New York
    address 192.168.102.60 ratio 4 // Los Angeles
    address 192.168.103.60 ratio 1 // Tokyo
  }
}


// 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"
    type vsb
    ratio 2
    preferred qos
    alternate ratio
    address 192.168.101.70 ratio 7
    address 192.168.102.60 ratio 2
  }

  pool {
    name "host_pool"
    type vsh
    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"
  pool_lbmode     ratio
  pool {
    name "pool_1"
    type vsb
    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"
    type vsh
    ratio 1
    preferred ratio
    address 192.168.104.50 ratio 2
    address 192.168.105.50 ratio 1
  }
}


// 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"
    type vsb
    ratio 2
    preferred leastconn
    alternate null
    fallback null
    address 192.168.101.50 ratio 2
    address 192.168.101.60 ratio 1
    address 192.168.101.70 ratio 1
  }

  pool {
    name "Los Angeles"
    type vsb
    ratio 1
    preferred leastconn
    alternate null
    fallback null
    address 192.168.102.50 ratio 3
    address 192.168.102.60 ratio 2
    address 192.168.102.70 ratio 1
  }

  pool {
    name "Tokyo"
    type vsb
    ratio 1
    preferred leastconn
    alternate null
    fallback return_to_dns
    address 192.168.103.50 ratio 3
    address 192.168.103.60 ratio 2
    address 192.168.103.70 ratio 1
  }
}


// Topological distribution and access control
topology {
  acl_threshold   1
  limit_probes    yes
  longest_match   yes

// Server       LDNS            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

}
```

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

To understand how cur_ values are used, you must first have a basic understanding of the 3DNS database.

The 3DNS database contains collected statistics and metrics. This collected information, and the specified load balancing mode, is used to determine how to distribute client requests. At each interval specified in the globals sync_db_interval sub-statement, the database is updated with a new configuration dump file, called */var/run/wideip.out*. The *wideip.out* file contains the most recent statistics, including cur_ values.

If both a cur_ value and an existing statistic or metric refer to the same thing, the cur_ value overwrites the existing information when *named* reads the */var/run/wideip.in* file as part of the database synchronization that a data copier performs each sync_db_interval seconds.

You may notice cur_ values in bigip, host, vs, path, or wideip definitions. Examples for each type of definition follow.

Example: bigip definition

```
bigip {
  // New York BIG/ip Controller
  address 192.168.101.40
  cur_packet_rate 139
  cur_ok 1
  [virtual server definitions]
}
```

In the above example, the cur_ values indicate the following.

Parameter	Description
cur_packet_rate	The number of packets per second sent during the last sample period.
cur_ok	The state of the specified BIG/ip Controller. The options are: 1 (Up), 2 (Down), 3 (Waiting), 4 (Alert), and 5 (Panic).

Example: host definition

```
host {
  // New York host
  address 192.168.105.40
  probe_protocol icmp
  prober 192.168.103.40 // Use the prober in Tokyo
  cur_ok 2
  [virtual server definitions]
}
```

In the preceding example, the `cur_` value indicates the following:

Parameter	Description
`cur_ok`	The state of the specified host machine. The options are: 1 (Up) and 2 (Down).

Example: vs definition

```
vs {
  address 192.168.102.50:80
  cur_serv_cnt 1
  cur_connections 0
  cur_picks 39
  cur_refreshes 783
}
```

In the above example, the `cur_` values indicate the following:

Parameter	Description
`cur_nodes_up`	The number of active servers serving the specified virtual server.
`cur_connections`	The number of connections to the specified virtual server.
`cur_picks`	The number of times the specified virtual server was returned by the 3DNS Controller.
`cur_refreshes`	The number of times the server and connection counts were refreshed with new data from a BIG/ip Controller.

Example: path definition

```
path {
  address 10.25.50.100 // LDNS
  cur_rtt 102382
  cur_completion_rate 10000
  cur_picks 239
  cur_accesses 302
}
```

In the above example, the `cur_` values indicate the following:

Parameter	Description
`cur_rtt`	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`	The percentage of completed packets versus lost packets, using this equation: [1 - (packets received / sent)] X 10000.
`cur_picks`	The number of times this path's data resulted in the corresponding BIG/ip Controller's virtual server being chosen for a connection. This only applies if a wide IP is doing dynamic load balancing (using path data).
`cur_accesses`	The number of times this path was considered when performing dynamic load balancing.

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]
}
```

In the above example, the `cur_` values indicate the following:

Parameter	Description
`cur_preferred`	The number of times the specified wide IP was resolved by the `preferred` load balancing mode.
`cur_alternate`	The number of times the specified wide IP was resolved by the `alternate` load balancing mode.
`cur_fallback`	The number of times the specified wide IP was resolved by the `fallback` load balancing mode.
`cur_returned_to_dns`	The number of times the specified wide IP couldn't 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.

◆ **Note**

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`.

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 tool 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 tool 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 tool 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] [-o] [-if <file_name>]  [-of <file_name>] \
   [-version] [-sf <file_name>] [-d] [-s] [-vl] [-picky]
```

The options for 3dparse include:

`-help`

Displays the list of available options.

`-o`

Writes the in-memory configuration to the *wideip.conf* file. The in-memory configuration is created by reading the input file and applying verification and validation.

`-if <file_name>`

Specifies a file name for the input file. If you don’t use this option, 3dparse uses the default input file, *wideip.conf*.

`-of <file_name>`

Specifies a file name for the output file.

`-version`

Displays the version information.

`-sf <file_name>`

Path for output status file. The default is *stdout*.

`-d`

Simulate an `ndc dumpdb` after parsing.

`-s`

Simulate data copier behavior when loading.

`-vl`

Turn on syslog verbosity and path loading.

`-picky`

Do not auto-correct any validation errors.

Example

The following example shows a 3dparse command. The bold typeface indicates the command entered.

```
bighost:~# 3dparse -o
3dparse: Initializing ...
3dparse: Parsing /etc/wideip.conf
3dparse: Dumping ./3dparse.out
3dparse: SUCCESS
```

watchdog-named

Use the *watchdog-named* utility to ensure that a version of *named* is always running on the 3DNS Controller.

If *watchdog-named* is running, do not manually start *named*. The 3DNS Controller does not prevent more than one *named* process from running simultaneously, and *watchdog-named* only monitors one *named* process at a time.

Because *watchdog-named* is not a daemon, start it as a background process.

watchdog-named performs the following functions:

- Starts and watches a new *named* process if *named* is not running when *watchdog-named* is started.
- Monitors any running *named* process.
- Starts a new *named* process if the watched *named* process stops.

- Keeps secure any dumped *named* core files by renaming the core file and adding a timestamp suffix. *watchdog-named* then compresses the core file.
- Presents an error message if you attempt to start more than one *watchdog-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 `-r` or `-s` arguments when you start *watchdog-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 that one directory command is found in *named.conf*, *watchdog-named* uses the last one it finds.

When your 3DNS Controller is using *watchdog-named*, you cannot use *ndc* to stop, start, or restart *named*. Instead, you must use 3ndc. See *3ndc*, on page D-5.

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 *watchdog-named* appears in the output.

A 3DNS Controller does not have to use *watchdog-named*. You can instead use *named* and *ndc*. See *named*, on page D-6, and *ndc*, on page D-8.

watchdog-named uses the following syntax:

```
watchdog-named [-c <path>] [-r <number>] [-s <number>]
```

The options for *watchdog-named* include:

`-c <path>`

Specifies the path for the *named.conf* file to use. The default is */etc/named.conf*.

`-r <number>`

Specifies the number of times *named* can be restarted before a warning is logged. The default is 10.

`-s <number>`

Specifies the number of seconds between restarts that is considered excessive. The default is 3600.

3ndc

3ndc allows the name server administrator to send various signals to the name server, or to restart it. *3ndc* is should be used in place of *ndc* on 3DNS Controllers that use *watchdog-named*.

Only use *3ndc* if *watchdog-named* is being used on your 3DNS Controller.

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* 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 set 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 *watchdog-named*, if it is not running. *watchdog-named* starts *named*. If a *named* process is already running, *watchdog-named* starts and watches the current *named* process.

stop

Stops *watchdog-named* and *named*, if they are running.

restart

Stops and restarts *watchdog-named* and *named*.

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.

ndc

The name daemon control interface command *ndc* allows a name server administrator to send signals to the name server. This section describes *ndc*; a sub-section describes the value that the 3DNS platform adds to the normal *ndc* functionality present in BIND.

◆ **WARNING**

Do not use ndc with watchdog-named. Instead, use 3ndc.

The syntax for ndc is as follows:

```
ndc directive [ ... ]
```

When you use *ndc*, you can specify directives. Directives are not required. The directives available for *3ndc* include:

`status`

Display the current status of *named* as shown by ps.

`dumpdb`

Write the 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. Use this directive with caution, as it sometimes starts two copies of *ndc*.

`stats`

Writes its statistics to */var/tmp/named.stats*. It 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. It uses the USR1 signal.

`notrace`

Sets its tracing level to zero, closing */var/tmp/named.run* if it is open. It uses the USR2 signal.

`querylog`

Toggles the query logging feature which, while on, results in a syslog 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 *named*, as long as it isn't already running.

`stop`

Stops *named*, if it is running.

`restart`

Stops and restarts *named*.

Signals and dump files: extending ndc

As mentioned above, the 3DNS Controller extends the functionality of *ndc* to send signals to the 3DNS Controller and dump data to the 3DNS Controller files.

To send signals to the 3DNS Controller name server (*named*), use one of the following commands:

```
kill -<signal code> `cat /var/run/named.pid`
```

or

```
ndc <signal function name>
```

The following signal codes are used by the 3DNS Controller in addition to the normal BIND functionality:

`HUP (name: restart)`

Restarts the name server. Use this signal to reread the *named.conf* and the *wideip.conf* files.

`INT (name: dumpdb)`

Dumps data metrics for wide IP addresses, BIG/ip Controllers, hosts, paths, and virtual servers in the following files, which are located in */var/run*:

- *3dns.sum*
- *3dns.paths*
- *3dns.ldns*
- *3dns.vs*
- *3dns.bigips*
- *3dns.hosts*
- *3dns.wips*
- *3dns.lbs*

These files correspond to the tables displayed in the F5 Configuration utility.

In addition, a memory representation of the 3DNS Controller is dumped to *wideip_dump.db* in *wideip.conf*-compliant format (C-like format).

◆ **Note**

The preceding information describes the low-level mechanics of how the 3DNS Controller administration tool obtains its information. This information can be useful for troubleshooting purposes.

`ABRT (name:stats)`

Dumps static information to */var/run/3dnsStats.log*.

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 *big3d* and *named* use the syslog daemon and all messages are written to the local2 facility.

To set up 3DNS Controller logging:

3. Add the following line to the */etc/syslog.conf* file.

   ```
   local2.err                          /var/log/3dns
   ```

   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.

4. Create an empty 3DNS Controller file in */var/log* by typing the following on the command line:

   ```
   % touch 3dns
   ```

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
```

5. 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 case insensitive.

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 didn't 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 <inverval> ]
```

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).

thttpd

The thttpd server is a simple, small, fast, and secure HTTP server. It is distributed and installed with the 3DNS Controller, and it supports the 3DNS Web Administration tool.

For more information on the thttpd server, see the following Web page: *www.acme.com/software/thttpd/.*

thttpd uses this syntax:

```
thttpd [-p <port>] [-d <dir>] [-r | -nor] [-u <user>] \
   [-c <cgipat>] [-t <throttles>] [-h <host>] [-l <logfile>]
```

Options for the thttpd server include:

-p

Specifies an alternate port number to listen on. The default is 80.

-d

Specifies a directory to chdir() to at startup.

-r

Performs a chroot() at initialization time, restricting file access to the program's current directory. If **-r** is the compiled-in default, **-nor** disables it.

-u

Specifies what user to switch to after initialization when started as root. The default is **nobody**.

-c

Specifies a pattern for CGI programs.

-t

Specifies a file of throttle settings

-h

Specifies a host name to bind to, for multi-homing. The default is to bind to all host names supported on the local machine.

-l

Specifies a file for logging. If no file is specified, thttpd logs via syslog.

Basic authentication

The version of thttpd that is installed with the 3DNS Controller includes the basic authentication feature, which is available as an option at compile time. If basic authentication is enabled, it uses a password file in a served directory, called *.htpasswd* by default. This file is formatted as the familiar colon-separated username/encrypted-password pair, with records delimited by new lines. The protection does not carry over to subdirectories. *htpasswd* is the name of the included utility program that helps create and modify *.htpasswd* files.

htpasswd uses this syntax:

```
htpasswd [-c ] passwordfile username
```

Using the above command sets a user's password in an httpd-style password file. The **-c** flag creates a new file.

Scripts

This section provides information on each script that is shipped with the 3DNS Controller. Most scripts correspond to items on the 3DNS Maintenance menu, which is shown on page 4-23. 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.

File location

All scripts are located in */usr/contrib/bin*, as are both data files. The data files are:

- **bigips.txt**
  This file consists of a list of the physical, external IP address of each BIG/ip Controller that is managed by the 3DNS Controller. The format is one IP address per line. If you have a BIG/ip redundant hardware system, the IP addresses of both BIG/ip machines are listed. You can edit this file by using the **Edit BIG/ip List** item on the 3DNS Maintenance menu.
- **3dns.txt**
  This file consists of a list of administration IP addresses of 3DNS Controllers. The format is one IP address per line. You can edit this file by using the **Edit 3DNS List** item on the 3DNS Maintenance menu. Note that you should not list the current 3DNS Controller's IP address in its own *3dns.txt* file.

You can use shell style (also known as Perl style) comments in both *bigips.txt* and *3dns.txt*. Shell style comments begin with a pound sign character (#) and are no longer than one line in length.

3dns_admin_start

The 3dns_admin_start script starts the Web Administration tool provided with your 3DNS Controller. For information on this tool, see Chapter 6, *Web Administration*.

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** item 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.*

The 3dns_auth script does the following:

1. 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-standby# 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
```

2. Appends 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.

To test that you have successfully generated the ssh key, use ssh to log into the data collector without a password:

`data collector#` **`ssh root@<ip-address-of-3DNS>`**

◆ **Note**

There may be cases where you have an existing ***identity.pub*** *file, but you want to perform the other tasks performed by 3dns_auth. In these cases, do not run the script again. Instead, perform the other tasks manually.*

3dns_dump

Without an argument, this script simply dumps the *named* cache and creates new versions of the files */var/3dns/etc/wideip.conf.static* and */var/3dns/etc/wideip.conf.dynamic*, using file */var/run/wideip.cmd*. If a *wideip.cmd* file already exists before the 3dns_dump script is called, *wideip.cmd* will temporarily be moved, and then restored afterward. This script prints out an error message if *named* does not respond to the signal to dump or read in the command file.

3dns_mode <conf | watch>

This script takes an argument (`conf` or `watch`) and returns a text string that displays the *wideip.conf* mode that the 3DNS Controller is currently using.

The `conf` argument determines which *wideip.conf* mode is currently running. This argument is also available on the 3DNS Maintenance menu as the **Display mode of wideip.conf** command. There are four different modes:

- **Initial**
  The */etc/wideip.conf* file is an plain file (not a link), and the 3DNS Controller has never been put into Static or Dynamic mode.
- **Static**
  The */etc/wideip.conf* file is actually a link to */var/3dns/etc/wideip.conf.static*.
- **Dynamic**
  The */etc/wideip.conf* file is actually a link to */var/3dns/etc/wideip.conf.dynamic*.
- **Unknown**
  The */etc/wideip.conf* file is missing, or is linked to an unknown file, or is otherwise corrupt.

The `watch` argument determines whether *watchdog-named* is currently active. The script returns `yes` or `no`.

An invalid argument to *3dns_mode* returns a `?`.

3dns_sync

The 3dns_sync script corresponds to the **Synchronize Configuration Data** item on the 3DNS Maintenance menu. This script distributes the *wideip.conf* file from the current 3DNS Controller to all other 3DNS Controllers that are listed in the *3dns.txt* file. This synchronizes the 3DNS Controller configuration on all specified 3DNS Controllers. Only use the script if you are certain that you want the same *wideip.conf* on all machines. Having the same *wideip.conf* on all machines may not be desirable in all cases.

3dns_web_passwd

The 3dns_web_passwd script corresponds to the **Change/Add Users for 3DNS Web Administration** item on the 3DNS Maintenance menu. This script secures the 3DNS administration Web site using basic authentication. This script lets you provide restricted or administrative access to the 3DNS Web Administration site 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 Administration site.

It is important to note that if you do not use this script, all users have access to the 3DNS administration Web site.

The first time you use this script to provide access for a user name and password, you block access for all other users. You can run this script again any time you need to provide access for another user.

big3d_check

The big3d_check script corresponds to the **Check big3d** item on the 3DNS Maintenance menu. This script checks that each BIG/ip Controller listed in the *bigips.txt* file is running the *big3d* utility.

big3d_install

The big3d_install script corresponds to the **Install and Start big3d** item on the 3DNS Maintenance menu. This script installs and starts the appropriate version of the *big3d* utility 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* process.

2. Uses a matrix file to determine which version of *big3d* 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* and named utilities 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*.

Configuring the big3d process

The syntax is:

```
big3d [options]
```

Option	Description
`-foreground`	Runs the process in the foreground rather than as a daemon.
`-help`	Lists the available options.
`-keyfile`	Specifies the location of the key file for encryption.
`-rxbufsize`	Sets the size of the receive socket buffer.
`-txbufsize`	Sets the size of the transmit socket buffer.
`-version`	Displays version information.

big3d_restart

The big3d_restart script corresponds to the **Restart big3d** item on the 3DNS Maintenance menu. This script stops and restarts the *big3d* utility on each BIG/ip Controller that is listed in the *bigips.txt* file.

dynamic_wideip

This script puts the 3DNS Controller into dynamic mode for *wideip.conf*. The script is also available on the 3DNS Maintenance menu as the **Use Dynamic wideip.conf** command.

The script first dumps the *named* cache; if the dump fails, the 3DNS Controller prompts you to choose whether to continue the script or exit the script. We recommend that you exit the script if this error occurs. Once the dump is complete, one of the following events happens:

- If you are switching the 3DNS Controller from Initial mode to Dynamic mode, the script backs up the */etc/wideip.conf* file to */var/3dns/etc/wideip.conf.ORIG*, and changes */etc/wideip.conf* to link to */var/3dns/etc/wideip.conf.dynamic*.

- If you are switching the 3DNS Controller from Static mode to Dynamic mode, the script simply changes */etc/wideip.conf* to link to */var/3dns/etc/wideip.conf.dynamic*. (In Static mode, the link points to */var/3dns/etc/wideip.conf.static*.)

◆ **Note**

Running this script while the system is already in dynamic mode is ineffective, and does not change the state of the system.

edit_wideip

The edit_wideip script corresponds to the **Edit 3DNS Configuration** item on the 3DNS Maintenance menu. This script opens the current *wideip.conf* file in pico and allows you to edit it.

In Initial mode, the script edits */etc/wideip.conf*. In either Dynamic or Static mode, the script first dumps the *named* cache; if the dump fails, the 3DNS Controller prompts the user to choose whether to continue the script or exit the script (we recommend that you exit the script if this error occurs). Once the dump is complete, the script opens */var/3dns/etc/wideip.conf.static* (even if in dynamic mode) for editing in ***pico*** or ***vi***. Once the edits are completed and you close the text editor, *wideip.conf.static* is read as a command to reload into *named*.

install_key and F5makekey

The install_key script corresponds to the **Generate and Copy F5 iQuery Encryption Key** item 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 (20 characters in length) for the Blowfish encryption algorithm.

print_3dvips

The print_3dvips script corresponds to the **Fetch BIG/ip Configuration** item on the 3DNS Maintenance menu. This script reads the list of defined BIG/ip Controllers in the *bigips.txt* file, then retrieves and saves a list of all the virtual servers owned by the listed BIG/ip Controllers. The print_3dvips script saves the list of virtual servers in a format that is acceptable by the 3DNS Controller and */etc/wideip.conf*.

The generated list is saved in a file called */etc/bigip.lst*, and is useful in configuring the `bigip` statement in your *wideip.conf* file. See page 4-5.

◆ **Note**

This script is not available in the international version of 3DNS Controller.

static_wideip

This script puts the 3DNS Controller into Static mode for *wideip.conf*. The script is also available on the 3DNS Maintenance menu as the **Use Static wideip.conf** command.

The script first dumps the *named* cache; if the dump fails, the 3DNS Controller prompts you to choose whether to continue the script or exit the script. We recommend that you exit the script if this error occurs. Once the dump is complete, one of the following events happens:

- If you are switching the 3DNS Controller from Initial mode to Static mode, the script backs up the */etc/wideip.conf* file to */var/3dns/etc/wideip.conf.ORIG*, and changes */etc/wideip.conf* to link to */var/3dns/etc/wideip.conf.static*.
- If you are switching the 3DNS Controller from Dynamic mode to Static mode, the script simply changes */etc/wideip.conf* to link to */var/3dns/etc/wideip.conf.static.* (In Dynamic mode, the link points to */var/3dns/etc/wideip.conf.dynamic*.)

◆ **Note**

Running this script while the system is already in Static mode does not change the state of the system.

E

BIND 8 Configuration 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.

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.

Statement	Description
`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.

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.

Syntax

```
acl <name> {
  address_match_list
};
```

include statement

The `include` statement inserts the specified file at the point where the `include` statement is encountered. It cannot be used within another statement, though, so a line such as the following is not allowed:

```
acl internal_hosts {"include internal_hosts.acl"}
```

Use `include` to break the configuration up into easily-managed chunks. For example, the following lines could be inserted at the top of a BIND configuration file in order to include ACL and key information:

```
include "/etc/security/keys.bind";
include "/etc/acls.bind";
```

Be careful not to type `#include`, as you would in a C program, because # is used to start a comment.

Syntax

```
include <path_name>
```

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 such 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 *src/bin/named/named-bootconf.pl*, a Perl script that is part of the BIND 8.1 source kit.

F

DNS Resource Records

What are resource records?

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.

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, records define 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

Type	Description
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

Index