BIG-IP® Reference Guide

version 4.2

MAN-0044-01

Product Version

This manual applies to version 4.2 of the BIG-IP® product family.

Legal Notices

Copyright

Trademarks

Export Regulation Notice

This product may include cryptographic software. Under the Export Administration Act, the United States government may consider it a criminal offense to export this product from the United States.

Export Warning

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

FCC Compliance

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

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

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

Canadian Regulatory Compliance

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

Standards Compliance

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

Acknowledgments

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

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

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

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

This product includes software developed by Adam Glass.

This product includes software developed by Christian E. Hopps.

This product includes software developed by Dean Huxley.

This product includes software developed by John Kohl.

This product includes software developed by Paul Kranenburg.

This product includes software developed by Terrence R. Lambert.

This product includes software developed by Philip A. Nelson.

This product includes software developed by Herb Peyerl.

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

This product includes software developed by Chris Provenzano.

This product includes software developed by Theo de Raadt.

This product includes software developed by David Muir Sharnoff.

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

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

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

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

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

This product includes software developed by Christos Zoulas.

This product includes software developed by Charles Hannum.

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

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

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

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

This product includes software developed by the Apache Group for use in the Apache HTTP server project (http://www.apache.org/).

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

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

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

Table of Contents

Introduction

1 BIG-IP Overview

2 Using the Setup Utility

7
bigpipe Command Reference

Introduction

- IMPORTANT HARDWARE INFORMATION
- Getting started
- Using the Administrator Kit
- What's new in version 4.2
- Learning more about the BIG-IP product family

IMPORTANT HARDWARE INFORMATION

References to hardware and upgrades contained in this document are specific to F5 Networks hardware products. For information concerning the initial deployment of your system, see the *Deployment Guide* that was shipped with your system. For in-depth Dell-specific hardware information, see the server documentation that is provided on the *Resource* CD and that shipped with your system if you ordered printed documentation.

References to hardware-specific features of the F5 Networks IP Application Switch, such as the spanning tree protocol and port mirroring, are not supported on Dell™ PowerEdge™ hardware.

Getting started

Before you start installing the BIG-IP, we recommend that you browse the ***BIG-IP Solutions Guide*** and find the load balancing solution that most closely addresses your needs. If the BIG-IP® unit is running the 3-DNS software module, you may also want to browse the ***3-DNS Administrator Guide*** to find a wide area load balancing solution. Briefly review the basic configuration tasks and the few pieces of information, such as IP addresses and host names, that you should gather in preparation for completing the tasks.

Once you find your solution and gather the necessary network information, turn back to the ***Configuration Worksheet*** and ***Hardware Orientation*** poster for hardware installation instructions, and then return to the ***BIG-IP Solutions Guide*** to follow the steps for setting up your chosen solution.

Choosing a configuration tool

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

The Setup utility

All users need to use the Setup utility (formerly known as First-Time Boot utility). This utility walks you through the initial system set up. You can run the Setup utility from the command line, or from a web browser. The Setup utility prompts you to enter basic system information including a root password and the IP addresses that will be assigned to the network interfaces. For more information, see Chapter 2 of this guide.

The Configuration utility

The Configuration utility is a web-based application that you use to configure and monitor the load balancing setup on the BIG-IP. Once you complete the installation instructions described in this guide, you can use the Configuration utility to perform the configuration steps necessary for your chosen load balancing solution. In the Configuration utility, you can also monitor current system performance, and download administrative tools such as the SNMP MIB or the SSH client. The Configuration utility requires Netscape Navigator version 4.7, or Microsoft Internet Explorer version 5.0 or later.

The bigpipe and bigtop command line utilities

The **bigpipe**™ utility is the command line counter-part to the Configuration utility. Using **bigpipe** commands, you can configure virtual servers, open ports to network traffic, and configure a wide variety of features. To monitor the BIG-IP, you can use certain **bigpipe** commands, or you can use the **bigtop**™ utility, which provides real-time system monitoring. You can use the command line utilities directly on the BIG-IP console, or you can run commands using a remote shell, such as the SSH client (encrypted communications only), or a Telnet client (for countries restricted by cryptography export laws). For detailed information about the command line syntax, see Chapter 7, *bigpipe Command Reference*.

Using the Administrator Kit

The BIG-IP Administrator Kit provides all of the documentation you need in order to work with the BIG-IP. The information is organized into the guides described below. The following printed documentation is included with the BIG-IP unit.

- **Hardware Orientation Poster**
  This poster includes information about the BIG-IP unit. It also contains important environmental warnings.
- **Configuration Worksheet**
  This worksheet provides you with a place to plan the basic configuration for the BIG-IP.

The following guides are available in PDF format from the CD-ROM provided with the BIG-IP. These guides are also available from the first Web page you see when you log in to the administrative web server on the BIG-IP.

- **BIG-IP Solutions Guide**
  This guide provides examples of common load balancing solutions. Before you begin installing the hardware, we recommend that you browse this guide to find the load balancing solution that works best for you.

- **BIG-IP Reference Guide**
  This guide provides detailed configuration information for the BIG-IP. It also provides syntax information for **bigpipe** commands, other command line utilities, configuration files, system utilities, and monitoring and administration information.
- **3-DNS Administrator and Reference Guides**
  If your BIG-IP includes the optional 3-DNS module, your administrator kit also includes manuals for using the 3-DNS module. The ***3-DNS Administrator Guide*** provides wide area load balancing solutions and general administrative information. The ***3-DNS Reference Guide*** provides information about configuration file syntax and system utilities specific to the 3-DNS module.

Stylistic conventions

To help you easily identify and understand important information, our documentation uses the stylistic conventions described below.

Using the solution examples

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

Identifying new terms

To help you identify sections where a term is defined, the term itself is shown in bold italic text. For example, a ***virtual server*** is a specific combination of a virtual address and virtual port, associated with a content site that is managed by a BIG-IP or other type of host server.

Identifying references to objects, names, and commands

We apply bold text to a variety of items to help you easily pick them out of a block of text. These items include web addresses, IP addresses, utility names, and portions of commands, such as variables and keywords. For example, with the **bigpipe pool <pool_name> show** command, you can specify a specific pool to show by specifying a pool name for the **<pool_name>** variable.

Identifying references to other documents

We use italic text to denote a reference to another document. In references where we provide the name of a book as well as a specific chapter or section in the book, we show the book name in bold, italic text, and the chapter/section name in italic text to help quickly differentiate the two. For example, you can find information about **bigpipe** commands in Chapter 7, *bigpipe Command Reference* of this guide.

Identifying command syntax

We show complete commands in bold Courier text. Note that we do not include the corresponding screen prompt, unless the command is shown in a figure that depicts an entire command line screen. For example, the following command shows the configuration of the specified pool name:

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

or

```
b pool <pool_name> show
```

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

Item in text	Description
****	Indicates that the command continues on the following line, and that users should type the entire command without typing a line break.
< >	Identifies a user-defined parameter. For example, if the command has **<your name>**, type in your name, but do not include the brackets.
|	Separates parts of a command.
[ ]	Indicates that syntax inside the brackets is optional.
...	Indicates that you can type a series of items.

Table Intro.1 *Command line syntax conventions*

Finding additional help and technical support resources

You can find additional technical information about this product in the following locations:

- **Release notes**
  Release notes for the current version of this product are available from the product web server home page, and are also available on the technical support site. The release notes contain the latest information for the current version, including a list of new features and enhancements, a list of fixes, and, in some cases, a list of known issues.
- **Online help**
  You can find help online in three different locations:
  - The web server on the product has PDF versions of the guides included in the Administrator Kit.
  - The web-based Configuration utility has online help for each screen. Simply click the **Help** button.

  - Individual **bigpipe** commands have online help, including command syntax and examples, in standard UNIX man page format. Simply type the command followed by the word **help**, and the BIG-IP displays the syntax and usage associated with the command.

- **Third-party documentation for software add-ons**
  The web server on the product contains online documentation for all third-party software, such as GateD.
- **Technical support through the World Wide Web**
  The Dell | Support website at **support.dell.com** provides the latest technical documentation.

What's new in version 4.2

The BIG-IP offers the following major new features in version 4.2, in addition to many smaller enhancements.

Support for the Controller and IP Application Switch platforms

This release includes support for both the BIG-IP Controller and the IP Application Switch™ hardware platforms.

The Setup utility

This release includes a new Setup utility for initially configuring your BIG-IP system. The Setup utility replaces the web-based and console-based First-Time Boot utility. For more information, see Chapter 2, *Using the Setup Utility*.

Enhanced pools support

This release contains several new attributes that you can assign to load-balancing pools. These new attributes include support for Session Initiation Protocol (SIP) Call-ID persistence and Windows Terminal Server (WTS) persistence, enhanced ability to redirect HTTP requests, the ability to insert client IP addresses into HTTP requests, and the ability to set specific Quality of Service (QoS) and Type of Service (ToS) levels within a packet. Furthermore, this release allows you to configure a pool to automatically disable a SNAT or NAT connection, or to bypass the load balancing of a connection by automatically forwarding the connection, using IP routing. In addition to using these new pool attributes, you can also specify a pool of multiple default gateways, used for handling administrative traffic such as SSH, telnet, FTP, and HTTPS connections. For more information, see the *Pools* section in Chapter 4, *Configuring the High-Level Network*.

New filter for rewriting HTTP redirections

This release provides an ISAPI filter, called **redirectfilter.dll**, which allows IIS servers running Netscape to rewrite HTTP redirections. Rewriting HTTP redirections helps to ensure that SSL connections remain on a secure channel. By installing this filter on your IIS server, you offload the task of rewriting HTTP redirections from your SSL Accelerator proxy to your IIS server. For more information, see *Rewriting HTTP redirection*, on page 4-41.

New global variables

Two new global variables, **open_failover_ports** and **self_conn_timeout**, are included in this release. The **open_failover_ports** variable allows you to restrict network failover traffic on specific VLANs. The **self_conn_timeout** variable acts as a tracking mechanism for UDP connections For more information, see Chapter 7, *bigpipe Command Reference*.

Enhanced rules support

With this release comes a number of new variables and operators, to enhance the ways that you can select pools for load balancing. Using rule statements, you can now select pools based on such criteria as the IP protocol of a packet, TCP/UDP port numbers, and QoS and ToS levels. A rule can also now balance traffic based on whether the client IP address is a member of a specific class. For SSL Accelerator proxies, you can use rules to rewrite HTTP redirection to ensure that traffic remains on an SSL-secured channel. For more information, see the *Rules* section in Chapter 4, *Configuring the High-Level Network*.

Enhanced support for virtual servers

This release contains a number of enhancements to the BIG-IP virtual server. First, you can now define multiple wildcard virtual servers instead of a single wildcard virtual server only. For information on configuring multiple wildcard virtual servers, see *Creating multiple wildcard servers*, on page 4-74. Secondly, you can configure an option known as dynamic connection rebinding, designed for those virtual servers that are load balancing transparent devices such as firewalls or routers. Dynamic connection rebinding causes any connections to a node address or service to be redirected to another node, if the original node transitions to a DOWN state. Finally, you can prevent a virtual server from sending a TCP reset when a connection is timed out. For more information, see the *Virtual Servers* section in Chapter 4, *Configuring the High-Level Network*.

SSL Accelerator proxy enhancements

This release includes several important enhancements to the SSL Accelerator proxy. For example, you can now configure options such as specifying ways for an SSL proxy to manage client certificates, inserting headers into HTTP requests, specifying ciphers and protocol versions, and configuring SSL session cache size and timeout values.

This release also supports the SSL-to-Server option, which allows you to re-encrypt traffic after it has been decrypted by the BIG-IP. Previously available on the IP Application Switch only, this feature is now available on the BIG-IP Controller platform also. Moreover, this feature has been enhanced in this release to further ensure the security of SSL connections between the proxy and the server. For a complete description of all new SSL Accelerator proxy options, see the *Proxies* section in Chapter 4, *Configuring the High-Level Network*.

Support for the nCipher FIPS 140-1 level 3 certified SSL cryptographic module

For BIG-IP Controller platforms, option is available to install a FIPS 140-1-certified cryptographic network module. The BIG-IP FIPS hardware option is specifically designed for processing SSL traffic within environments that require FIPS 140-1 Level 3 compliant solutions. It comes with the FIPS 140-1 level 3 certified PCI based encryption processing module, attached smart card reader, and 5 smart cards. This product can be installed in any BIG-IP Controller platform (D35) that has BIG-IP software version 4.2 and is authorized by your vendor. For more information, see *Configuring FIPS 140 Security World on the BIG-IP* in the Documentation section of the Software and Documentation CD.

Enhanced support for Secure Network Address Translations (SNATs)

In previous releases, BIG-IP allowed you to automatically map VLANs to translation IP addresses during SNAT creation. In this release, you can now use this automapping feature not only for VLANs, but for one or more individual IP addresses. For more information, see the *Address Translation: NATs, SNATs, and IP Forwarding* section in Chapter 4, *Configuring the High-Level Network*.

Enhanced interface statistics

This release features enhanced statistics for BIG-IP interfaces. The following state information and statistics are now available: MTU, Speed, MAC address, packets in, errors in, packets out, errors out, collisions, dropped packets, bits in, bits out. Previously available on the IP Application Switch, this feature is new for the BIG-IP Controller platform. For more information, see Chapter 11, *Monitoring and Administration*.

Health monitor enhancements

In addition to the standard SNMP health monitor template included in BIG-IP, this release now includes a second SNMP template, which allows users to collect data on elements other than CPU, disk, and memory usage. For more information, see the *Health Monitors* section in Chapter 4, *Configuring the High-Level Network*.

Support for LDAP and RADIUS logins

With this release, BIG-IP can now authenticate SSH users by way of an LDAP or a RADIUS server. For information on configuring this feature, see *To configure RADIUS login support*, on page 12-14 and *Configuring LDAP login support*, on page 12-15.

Enhanced system logging

System logging in this release provides more detailed information, such as **up** or **down** status for nodes. For more information, see Chapter 11, *Monitoring and Administration*.

Web-based Configuration utility enhancements

This release includes a number of improvements to the web-based Configuration utility. All new features for this release are supported by the Configuration utility.

Learning more about the BIG-IP product family

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

- **The BIG-IP**
  The BIG-IP HA, HA+, and 5000 software provides the full suite of local area load balancing functionality. The BIG-IP unit also has an optional 3-DNS software module which supports wide-area load balancing.
- **The BIG-IP e-Commerce Controller**
  The BIG-IP e-Commerce Controller uses SSL acceleration technology to increase the speed and reliability of the secure connections that drive e-commerce sites.

- **The BIG-IP special purpose products**
  The special purpose BIG-IP provides the ability to choose from three different BIG-IP feature sets. When you run the Setup utility, you specify one of three types:
  - **The BIG-IP Load Balancer**
    The BIG-IP Load Balancer provides basic load balancing features.
  - **The BIG-IP FireGuard**
    The BIG-IP FireGuard provides load balancing features that maximize the efficiency and performance of a group of firewalls.
  - **The BIG-IP Cache Controller**
    The BIG-IP Cache Controller uses content-aware traffic direction to maximize the efficiency and performance of a group of cache servers.

1

BIG-IP Overview

- Introduction
- What is a BIG-IP?
- Configuration
- Monitoring and administration
- The BIG-IP user interface

Introduction

This chapter provides a brief overview of the BIG-IP software and the configuration and monitoring tasks associated with it as an introduction to the chapters that follow. (For an overview of BIG-IP functionality with sample solutions, refer to Chapter 1 of the ***BIG-IP Solutions Guide***.)

This chapter is organized as follows:

- What is a BIG-IP?
- Configuring the BIG-IP
- Monitoring the BIG-IP
- The BIG-IP user interface

What is a BIG-IP?

The BIG-IP is an Internet device used to implement a wide variety of load balancing and other network traffic solutions, including intelligent cache content determination and SSL acceleration.

Figure 1.1 shows the most basic kind of BIG-IP configuration. In it, the unit sits between a router and an array of content servers, and load balances inbound Internet traffic across those servers. (For an introduction to more complex solutions, including load balancing of outbound traffic across firewalls and routers, refer to the ***BIG-IP Solutions Guide***, Chapter 1, *Overview*.)

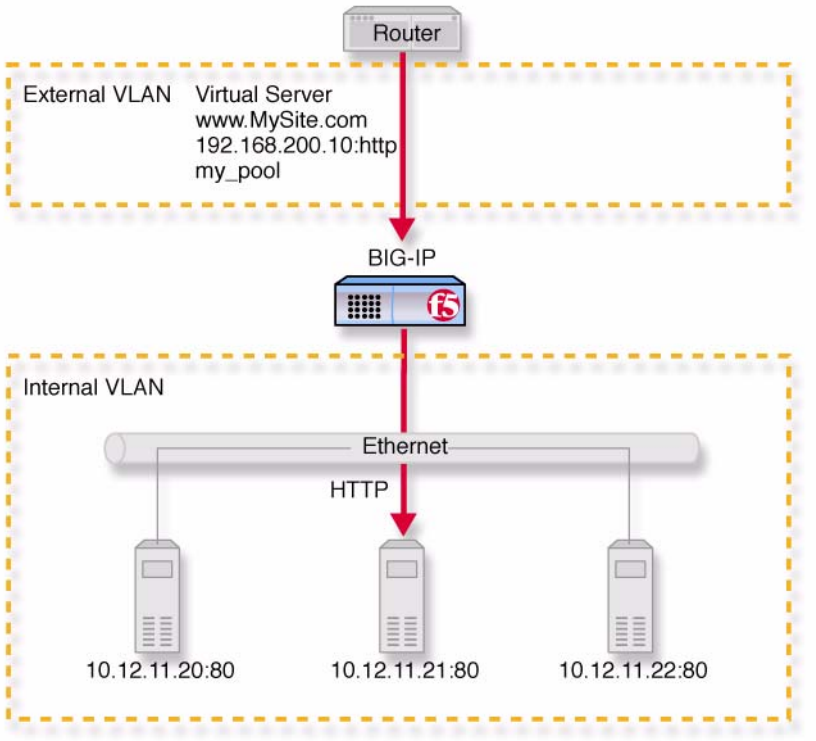

Figure 1.1 *A basic configuration*

Insertion of the BIG-IP, with its minimum of two interfaces, divides the network into an external VLAN and an internal VLAN. (Both VLANs can be on a single IP network, so that inserting the BIG-IP does not require you to change the IP addressing of the network.) The nodes on the external VLAN are routable. The nodes on the internal VLAN, however, are hidden behind the BIG-IP. What appears in their place is a user-defined ***virtual server***. It is this virtual server that receives requests and distributes them among the physical servers, which are now members of a ***load-balancing pool***.

The key to load balancing through a virtual server is address translation, and setting the BIG-IP address as the default route. By default, the virtual server translates the destination address of the incoming packet to that of the network device it load balances to, making it the source address of the reply packet. The reply packet returns to the BIG-IP as the default route, and the BIG-IP translates its source address back to that of the virtual server. (For outbound traffic, address translation can be modified or disabled to give internal nodes visibility to the Internet.)

Like the physical network itself, you can add software entities like virtual servers and load balancing pools to the BIG-IP, and any properties associated with them (like load balancing methods for pools). Adding hardware and software components to the BIG-IP is referred to collectively as ***configuration***.

Configuration

Configuration is setting up the BIG-IP to perform its load balancing and other functions on an ongoing basis. You configure the BIG-IP when it is first installed, and later as required by changing needs or changes in the network itself. For convenience, the BIG-IP configuration can be broken into the following components:

- Hardware configuration
- Base network configuration
- High-level network configuration

Figure 1.2 shows how these three kinds of configuration relate to one another.

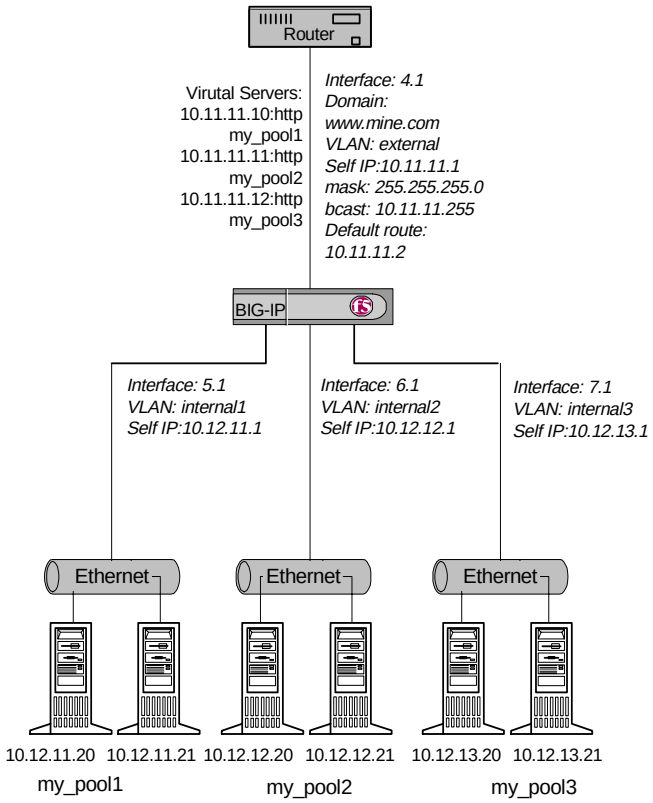

Figure 1.2 *Hardware configuration with base and high-level networks superimposed.*

Hardware configuration

The hardware configuration includes all physical devices and connections in Figure 1.2, in other words, it includes the entire physical network. In this case, it consists of a BIG-IP with four interfaces, one external and three internal, with each internal interface having its own Ethernet connecting to two physical servers. Solution-specific hardware configuration is provided in the ***BIG-IP Solutions Guide***.

Base network configuration

The base network consists of the BIG-IP interfaces and the domain names, self IP addresses, VLANs, and optional trunks that are built on them. Figure 1.2 shows this as italicized text. (In the example, the three internal interfaces are assigned to three separate VLANs, each with its own self IP address including netmask and broadcast address. If this were a BIG-IP redundant system, there would be additional floating self IP addresses for

sharing.) When you run the Setup utility as the last part of your initial hardware installation and fill in the required fields, you are configuring the base network. After you complete the Setup utility, you have, at the minimum, the two default VLANs (**external** and **internal**), domain names, and self IP addresses (both one true and floating as required) with netmask and broadcast addresses. This base configuration enables you, among other things, to access the BIG-IP from a remote host using SSH or HTTPS and in this way gain access to both the command line interface and the graphic Configuration utility.

At this point you may want to further configure the base network by changing settings, segmenting the network into more interface-group VLANs, adding VLANs with tagged interfaces, creating additional floating self IP addresses, and performing link aggregation. These additional configurations are solution-dependent and can be extensive, particularly if you have more than two interfaces on your default internal VLAN. (If, for example, you were hosting three customers, as in Figure 1.2, but were using a single interface with an external switch, you would need to segment what was originally the default internal VLAN into three separate tagged VLANs.)

You may also re-run the Setup utility in its entirety or using its various sub-utilities. For more information on these base configuration utilities, see Chapter 2, *Using the Setup Utility*.

High-level network configuration

Once a base network exists and you have administrative access to the BIG-IP and at least a default VLAN assignment for each interface, the next step is to configure a network for the web servers to be load balanced. Figure 1.2 shows the high-level network in non-italicized text. The network includes the server nodes, the pools containing those nodes, and the virtual servers that represent the pools to the client.

Just as the base network is built on the BIG-IP interfaces, the high-level network is built on the load balancing pool. Until there is a pool, there are no nodes to load balance. Once a pool exists, nodes come into existence as members of that pool and can receive traffic through a virtual server. The high-level network also includes the properties attaching to pools, virtual servers, and nodes such as persistence (a pool property), and any pool selection criteria as expressed in a ***rule***. The high-level network can also include proxies for SSL and akamaization, NATs, SNATs, and health monitor associations for specific nodes or all nodes.

For detailed information on configuration refer to Chapter 4, *Configuring the High-Level Network*, Chapter 7, *bigpipe Command Reference*, and Chapter 8, *Configuring SNMP*.

Global settings and filters

Global settings and filters are part of the configuration that belong to neither the base network nor the high-level network.

Global settings are settings that are system wide rather than applicable only to specific objects. Global settings are documented in Chapter 4, *Configuring the High-Level Network*, and under the **bigpipe global** command in Chapter 7, *bigpipe Command Reference*.

Filters include IP and Rate filters, and are covered in Chapter 5, *Configuring Filters*.

Monitoring and administration

Monitoring and administration refer to the day-by-day tasks of observing traffic, gathering statistics, adding users, and removing and returning items to service. Various utilities provide statistics in a variety of formats and may be global or specific to certain elements of the network, such as virtual servers, nodes, NATs, SNATs, or services. Monitoring and administration is covered in Chapter 11, *Monitoring and Administration*, and Chapter 8, *Configuring SNMP*.

The BIG-IP user interface

User interface to the BIG-IP consists primarily of the web-based Configuration utility and the command line interface **bigpipe**.

The Configuration utility

The Configuration utility resides in the BIG-IP internal web server. You can access it through the administrative interface on the BIG-IP using Netscape Navigator version 4.7, or Microsoft Internet Explorer version 5.0, or later. (Netscape Navigator version 6.0 is not supported.)

Figure 1.3 shows the Configuration utility as it first appears, displaying the **Network Map** with any existing nodes and virtual servers. The Configuration utility thus provides an instant overview of your high-level network as it is currently configured. (You can view the Base Network by clicking **Network** on the navigation pane.)

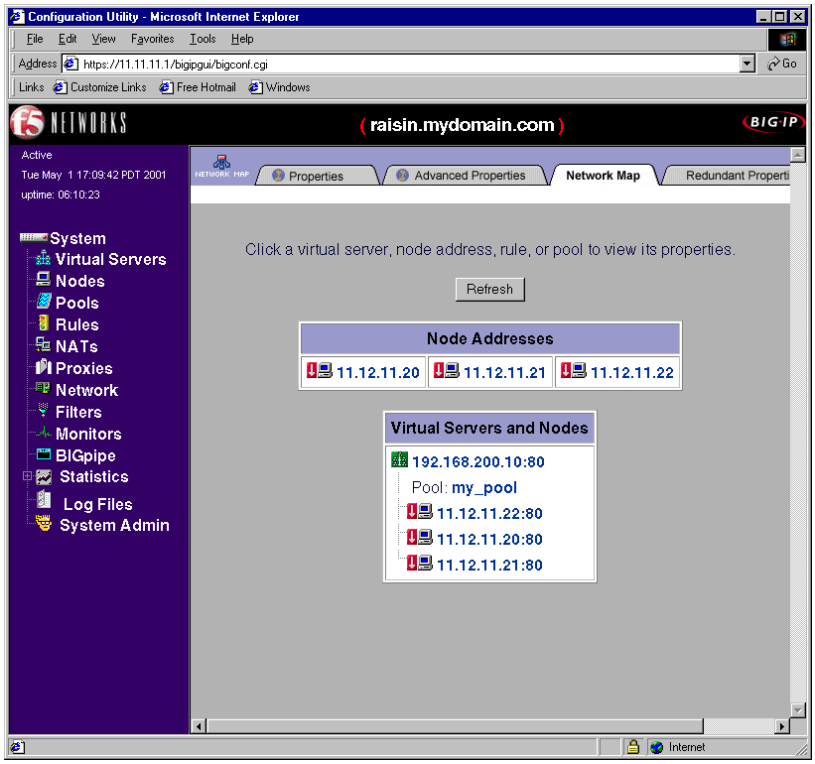

Figure 1.3 *Configuration utility System screen*

The left pane of the screen, referred to as the ***navigation pane***, contains links to **Virtual Servers**, **Nodes**, **Pools**, **Rules**, **NATs**, **Proxies**, **Network**, **Filters**, and **Monitors**. These screens appear in the right pane. The navigation pane also contains links to screens for monitoring and system administration (**Statistics**, **Log Files**, and **System Admin**).

As an example of using the Configuration utility, suppose you wanted to create a pool. You would click **Pools** to open the Pools screen, then click the **Add** (+) button to open the Add Pool screen, as shown in Figure 1.4.

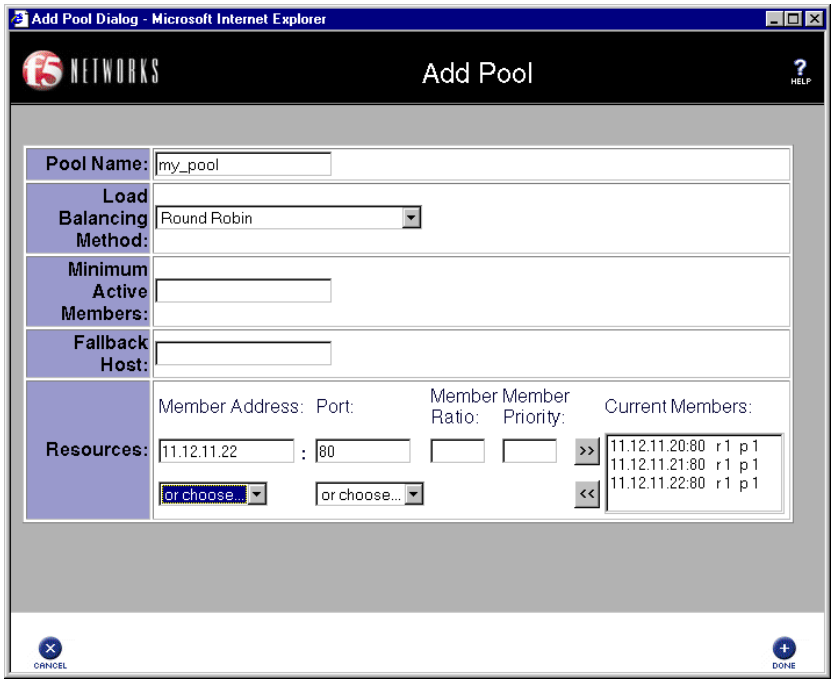

Figure 1.4 *Add Pool screen*

The Add Pool screen contains fields for all the attributes you can configure for the pool.

The bigpipe command line interface

You can access the command line interface **bigpipe** on a BIG-IP with connections for a monitor and keyboard. For a system without a monitor and keyboard attached (headless), like the IP Application Switch, you can access **bigpipe** through an SSH shell from a remote administrative host.

To give an example of a configuration using the **bigpipe** command line utility, the same pool shown in Figure 1.4 in the Add Pool screen would be configured at the command line as follows:

```
b pool my_pool { member 11.12.11.210:80 member 11.12.11.21:80 member 11.12.11.22:80 }
```

(You can use **b** or **bp** as shorthand for **bigpipe**.) For convenience, long commands like this can be entered using backslash breaks:

```
b pool my_pool { \
member 11.12.11.20:80 \
member 11.12.11.21:80 \
member 11.12.11.22:80 }
```

The bigip.conf file

Regardless of how a pool, virtual server, proxy or other object is configured, whether you use the Configuration utility or **bigpipe**, it is entered into the configuration file **/config/bigip.conf**. This produces an entry in that file like the one shown in Figure 1.5. As a third configuration option, you can also edit this file directly using a text editor like **vi** or **pico.**

```
pool my_pool {
       member 11.12.11.20:80
       member 11.12.11.21:80
       member 11.12.11.22:80
       }
```

Figure 1.5 *Pool definition in* ***bigip.conf***

When you run the Setup utility, the objects created in the base network are placed in a separate file of the same format, **/config/bigip_base.conf**.

2

Using the Setup Utility

- Creating the initial software configuration with the Setup utility
- Connecting to the BIG-IP for the first time
- Using the Setup utility for the first time
- Running the Setup utility after creating the initial software configuration

Creating the initial software configuration with the Setup utility

Once you install and connect the hardware, the next step in the installation process is to turn the system on and run the Setup utility. The Setup utility defines the initial configuration settings required to install the BIG-IP into the network. You can run the Setup utility remotely from a web browser, or from an SSH or Telnet client, or you can run it directly from the console.

Before you connect to the unit, we recommend that you gather the list of information outlined in the configuration worksheet provided with the BIG-IP. Note that the screens you see are tailored to the specific hardware and software configuration that you have. For example, if you have a stand-alone system, the Setup utility prompts you to skip the redundant system screens.

Once you have configured the base network elements with the Setup utility, you might want to further enhance the configuration of these elements. For additional information about these configuration tasks, see Chapter 3, *Additional Base Network Configuration.*

Connecting to the BIG-IP for the first time

The Setup utility prompts you to enter the same information, whether you run the utility from a web browser, or from the command line. When the utility completes we recommend that you reboot the unit. This automatically removes the default IP address and root password provided specifically for the purposes of running the Setup utility remotely. The BIG-IP replaces the default IP address and root password with the password and IP addresses that you define while running the utility.

Running the utility from the console or serial terminal

Before you can run the Setup utility from either the console or a serial terminal, you must first log in. Use the following default user name and password to log in.

Username: **root**

Password: **default**

After you log in, you can start the utility directly from the console or serial terminal by typing the command **config**. Once you complete the utility, we recommend that you reboot the BIG-IP. The BIG-IP automatically reboots if you are running the utility for the first time from the console.

◆ Note

If you want to set up a terminal connection directly to the BIG-IP, see ***Using a serial terminal with the BIG-IP****, on page 12-11.*

Running the Setup utility remotely

You can run the Setup utility remotely only from a workstation that is on the same LAN as the unit. To allow remote connections for the Setup utility, the BIG-IP comes with two pre-defined IP addresses, and a pre-defined root password. The default root password is **default**, and the preferred default IP address is **192.168.1.245**. If this IP address is unsuitable for your network, the BIG-IP uses an alternate IP address, **192.168.245.245**. However, if you define an IP alias on an administrative workstation in the same IP network as the BIG-IP, the unit detects the network of the alias and uses the corresponding default IP address.

Once the utility finishes and the system reboots, these default IP addresses are replaced by the information that you entered in the Setup utility.

Setting up an IP alias for the default IP address before you start the unit

You must set up an IP alias for your remote workstation before you turn on the unit and start the Setup utility. The remote workstation must be on the same IP network as the unit. If you add this alias prior to booting up the BIG-IP, the unit detects the alias and uses the corresponding address.

To set up an IP alias for the alternate IP address

The IP alias must be in the same network as the default IP address you want the BIG-IP to use. For example, on a UNIX workstation, you might create one of the following aliases:

- If you want the unit to use the default IP address **192.168.1.245**, then add an IP alias to the machine you want to use to connect to the unit using the following command:

  ```
  ifconfig exp0 add 192.168.1.1
  ```

- If you want to use the default IP address **192.168.245.245**, then add an IP alias such as:

  ```
  ifconfig exp0 add 192.168.245.1
  ```

On Microsoft Windows® or Windows NT® machines, you must use a static IP address, not DHCP. Within the network configuration, add an IP alias in the same network as the IP in use on the unit. For information about adding a static IP address to a Microsoft Windows operating system, please refer to your vendor's documentation.

Determining which default IP address is in use

After you configure an IP alias on the administrative workstation in the same IP network as the BIG-IP and you turn the system on, the BIG-IP sends ARPs on the internal VLAN to see if the preferred **192.168.1.245** IP address is in use. If the address is appropriate for your network and is currently available, the BIG-IP assigns it to the internal VLAN. You can immediately use it to connect to the unit and start the Setup utility.

If the alternate network is present on the LAN, **192.168.245.0/24**, or if the node address **192.168.1.245** is in use, then the BIG-IP assigns the alternate IP address **192.168.245.245** to the internal VLAN instead.

Starting the utility from a web browser

When you start the utility from a web browser, you use the selected default IP address as the application URL.

To start the Setup utility in a web browser

1. Open a web browser on a workstation connected to the same IP network as the internal VLAN of the unit.
2. Type the following URL, where **<default IP>** is the IP address in use on the BIG-IP internal VLAN.
   https://<default IP>
3. At the login prompt, type **root** for the user name, and **default** for the password.
   The Configuration Status screen opens.
4. On the Configuration Status screen, click **Start Wizard**.
5. Fill out each screen using the information from the Setup utility configuration list. After you complete the Setup utility, the BIG-IP reboots and uses the new settings you defined

◆ Note

You can rerun the Setup utility from a web browser at any time by clicking the Setup utility link on the welcome screen.

Starting the utility from the command line

You can run the command line version of the Setup utility from the console or serial terminal, or from a remote SSH client or from a Telnet client.

To start the Setup utility from the console

1. At the login prompt, type **root** for the user name, and **default** for the password.
2. At the BIG-IP prompt, type the following command to start the command-line based Setup utility.

   ```
   config
   ```

3. Fill out each screen using the information from the Configuration worksheet. After you complete the Setup utility, the BIG-IP reboots and uses the new settings you defined.

To start the Setup utility from the command line from a remote administrative workstation

1. Start an SSH client on a workstation connected to the same IP network as the internal VLAN of the unit. (See *Downloading the SSH client to your administrative workstation*, on page 12-3, for information on downloading the SSH client from the BIG-IP.)
2. Type the following command, where **<default IP>** is the IP address in use on the BIG-IP internal VLAN.

   ```
   ssh <default IP>
   ```

3. At the login prompt, type **root** for the user name, and **default** for the password.
4. At the BIG-IP prompt, type the following command to start the command-line based Setup utility.

   ```
   config
   ```

5. Fill out each screen using the information from the Configuration worksheet. After you complete the Setup utility, reboot the BIG-IP by typing the following command:

   ```
   reboot
   ```

◆ Note

*You can rerun the Setup utility at any time using the **config** command.*

Using the Setup utility for the first time

The following sections provide detailed information about the settings that you define in the Setup utility.

Keyboard type

Select the type of keyboard you want use with the BIG-IP. The following options are available:

- Belgian
- Bulgarian MIK
- French
- German
- Japanese - 106 key
- Norwegian
- Spanish
- Swedish

- US + Cyrillic
- US - Standard 101 key (default)
- United Kingdom

Product selection

If you are configuring a BIG-IP Cache Controller, BIG-IP FireGuard, or BIG-IP LB Controller, you must now select one of these three as your product. When you have made your selection, the features supported by that product will be enabled.

◆ Note

*You may change your product selection at a later time by running command line version of the Setup utility and selecting the **Select type of BIG-IP option**.*

Once you have configured your system based on one of the three product selections (BIG-IP Cache Controller, BIG-IP FireGuard, or BIG-IP LB Controller), changing the product selection will most likely invalidate that configuration. Therefore, you need to change and update your configuration after you have rebooted the system under the new product selection.

Root password

A root password allows you command line administrative access to the BIG-IP system. We recommend that the password contain a minimum of 6 characters, but no more than 32 characters. Passwords are case-sensitive, and we recommend that your password contain a combination of upper- and lower-case characters, as well as numbers and special characters for example, **!@#$%^&***. Once you enter a password, the Setup utility prompts you to confirm your root password by typing it again. If the two passwords match, your password is immediately saved. If the two passwords do not match, the Setup utility provides an error message and prompts you to re-enter your password.

Host name

The host name identifies the BIG-IP itself. Host names must be fully qualified domain names (FQDNs). The host portion of the name must start with a letter, and must be at least two characters. The name, or host part of the name, must be 39 characters or less. The label part of the name must be 63 characters or fewer. For example:

```
<host 39 characters or less>.<label 63 characters or less>.net
```

Configuring a default gateway pool

If a BIG-IP does not have a predefined route for network traffic, the unit automatically sends traffic to the pool that you define as the default gateway pool. You can think of the default gateway pool as a pool of default routes. Typically, a default gateway pool is set to two or more gateway IP addresses. If you type more than one default gateway IP address, the additional gateways provide high availability for administrative connections. If a gateway in the default gateway pool becomes inactive, existing connections through the inactive gateway are routed through another gateway in the default gateway pool. If you type one IP address, no pool is created and the address is entered as the default route.

All default gateway IP addresses you add to the default gateway pool must be in the same IP network as the BIG-IP.

Redundant system settings

There are two types of settings you need to define for redundant systems: unit IDs, and fail-over IP addresses.

Unit IDs

The default unit ID number is **1**. If this is the first unit in the redundant system, use the default. When you configure the second unit in the system, type **2**. These unit IDs are used for active-active redundant configuration.

Choosing a fail-over IP address

A fail-over IP address is the IP address of the unit which will take over if the current unit fails. Type in the IP address configured on the internal interface of the other BIG-IP in the redundant pair.

Setting the interface media type

Configure media settings for each interface. The media type options depend on the network interface card included in your hardware configuration. The Setup utility prompts you with the settings that apply to the interface installed in the unit. The BIG-IP supports the following types:

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

◆ Note

*For best results, choose the **auto** setting. In some cases, devices configured for the auto media are incompatible, and the proper duplex setting will not be negotiated. In these cases you may need to set the media settings to the same speed and duplex on this device and the corresponding switch or host. Check your switch or hub documentation for this information.*

The configuration utility lists only the network interface devices that it detects during system boot. If the utility lists only one interface device, the network adapter may have come loose during shipping. Check the LED indicators on the network adapters to ensure that they are working and are connected.

Configuring VLANs and IP addresses

You can create a new VLAN or use the default internal and external VLANs to create the BIG-IP configuration.

Determine whether you want to have security turned **on** for a VLAN, or **off** for the VLAN. Then, type the IP address settings for the VLAN. The IP address settings include:

- Security settings
- IP address, netmask, and broadcast
- Floating self IP address, netmask, and broadcast

We recommend that you set the floating self IP address as the default route for target devices, such as servers. The floating self IP address is owned by the active unit in an active/standby configuration.

◆ Note

The IP address of the external VLAN is not the IP address of your site or sites. The IP addresses of the sites themselves are specified by the virtual IP addresses associated with each virtual server you configure.

Assigning interfaces to VLANs

After you configure the VLANs you want to use on the BIG-IP, you can assign interfaces to the VLANs. If you use the default internal and external VLANs, we recommend that you assign at least one interface to the external VLAN, and at least one interface to the internal VLAN. The external VLAN is the one on which the BIG-IP receives connection requests. The internal VLAN is typically the one that is connected to the network of servers, firewalls, or other equipment that the BIG-IP load balances.

Associating the primary IP address and VLAN with the host name

After you assign interfaces to VLANs, you can choose one VLAN/IP address combination as the primary IP address to associate with the unit host name.

Configuring remote web server access

The BIG-IP web server provides the ability to set up remote web access on each VLAN. When you set up web access on a VLAN, you can connect to the web-based configuration utility through the VLAN. To enable web access, specify a fully qualified domain name (FQDN) for each VLAN. The BIG-IP web server configuration also requires that you define a user ID and password. If SSL is available, the configuration also generates authentication certificates.

The Setup utility guides you through a series of screens to set up remote web access.

- The first screen prompts you to select the VLAN you want to configure for web access. After you select an interface to configure, the utility prompts you to type a fully qualified domain name (FQDN) for the interface. You can configure web access on one or more interfaces.
- After you configure the interface, the utility prompts you for a user name and password. After you type a user name and password, the utility prompts you for a vendor support account. The vendor support account is not required.
- The certification screen prompts you for country, state, city, company, and division.

If you ever change the IP addresses or host names on the BIG-IP interfaces, you must reconfigure the BIG-IP web server and the portal to reflect your new settings.

You can also add users to the existing password file, change a password for an existing user, or recreate the password file, without actually repeating the remote web server configuration process.

If you have modified the remote web server configuration outside of the configuration utility, be aware that some changes may be lost when you run the Setup utility. This utility overwrites the ***httpd.conf*** *file and* ***openssl.conf****.*

Setting the time zone

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

Configuring the DNS proxy forwarding settings

This option is only available if you do not have the 3-DNS module installed. You need to complete this step only if you want machines inside your BIG-IP managed network to use DNS servers outside of that network (for example, for reverse DNS lookup from a web server).

Specify the DNS name server and domain name for DNS proxy forwarding by the BIG-IP. For more information on DNS proxy forwarding see *Configuring DNS on the BIG-IP*, on page 12-8.

If you have the 3-DNS module installed, please refer to the 3-DNS documentation for more information about configuring DNS.

Configuring remote administrative access

After you configure remote web access, the Setup utility prompts you to configure remote command line access. On most BIG-IP units, the first screen you see is the Configure SSH screen, which prompts you to type an IP address for SSH command line access. If SSH is not available, you are prompted to configure access through Telnet and FTP instead.

When you configure shell access, the Setup utility prompts you to create a support account for that method. You can use this support account to provide a support engineer access to the BIG-IP.

When the Setup utility prompts you to enter an IP address for administration, you can type a single IP address or a list of IP addresses, from which the BIG-IP will accept administrative connections (either remote shell connections, or connections to the web server on the BIG-IP). To specify a range of IP addresses, you can use the asterisk (*****) as a wildcard character in the IP addresses.

The following example allows remote administration from all hosts on the **192.168.2.0/24** network:

```
192.168.2.*
```

◆ Note

For administration purposes, you can connect to the BIG-IP floating self IP address, which always connects you to the active unit in an active/standby redundant system. To connect to a specific unit, connect directly to the IP address of that BIG-IP.

Configuring remote access for noncrypto-enabled versions of the system

The Telnet and FTP configuration options are presented only if you do not have a full crypto-enabled version of the BIG-IP. If you have a crypto-enabled version of the software, you are prompted to configure SSH.

Configuring Telnet

Use this option to configure the Telnet server on a BIG-IP only. The Setup utility prompts you to configure each service independently. This allows you to enable Telnet.

The utility prompts you for a configuration address for each service from which administrators may access the BIG-IP. You can use wildcard characters (*****) to include all addresses from a specific part of the network. This utility also prompts you to create a support account for access by technical support.

If **inetd** is not currently configured, this utility configures **inetd** for the requested services. If the ports for Telnet are closed, this utility opens the ports to permit Telnet connections to the BIG-IP.

Configuring FTP

Use this option to configure FTP on the BIG-IP. The Setup utility prompts you for an IP address from which administrators may access the BIG-IP with FTP. You can use wildcard characters (*****) to include all addresses from a specific part of the network. This utility also prompts you to create a support account for access by technical support.

If the service port for FTP is closed, this utility opens the service port to permit FTP connections to the BIG-IP.

Configuring NTP support

You can synchronize the time on the unit to a public time server by using Network Time Protocol (NTP). NTP is built on top of TCP/IP and assures accurate, local timekeeping with reference to clocks located on the Internet. This protocol is capable of synchronizing distributed clocks, within milliseconds, over long periods of time. If you choose to enable NTP, make sure UDP port **123** is open in both directions when the unit is behind a firewall.

Configuring NameSurfer for zone file management

If you have the 3-DNS module installed, you can configure NameSurfer to handle DNS zone file management. We strongly recommend that you configure NameSurfer to handle zone file management by selecting NameSurfer to be the master on the unit. If you select NameSurfer as the master, NameSurfer converts the DNS zone files on the system, becomes the

authoritative DNS, and automatically processes changes and updates to the zone files. (You can access the NameSurfer application directly from the Configuration utility for the 3-DNS module.)

Running the Setup utility after creating the initial software configuration

You normally run the Setup utility when the system is first installed as part of the installation procedure. However, you can also use the command line Setup utility to change existing settings at any time. This section describes running the Setup utility to change settings after you run it initially.

To run the Setup utility from the command line, type in the following command:

```
config
```

After you complete the initial configuration, the Setup utility presents a menu of individual configuration options.

The Setup utility menu is divided into two different sections. The Setup utility includes the following required configuration options:

- Set the default gateway pool
- Configure VLANs and networking
- Set host name
- Set the root password
- Configure web servers
- Specify type of BIG-IP (some versions of BIG-IP)
- Steps for redundant systems (redundant systems only)

The following configuration selections are optional:

- Configure DNS
- Configure FTP
- Set keyboard type
- Define time servers
- Configure NameSurfer (3-DNS module)
- Initialize the iControl portal
- Configure RSH
- Configure SSH
- Configure telnetd
- Set time zone

```
lqq I N I T I A L   S E T U P   M E N U qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
x                                                                                x
x                                                                                x
x    Choose the desired configuration function from the list below.            x
x                                                                                x
x                                                                                x
x    (A) All configuration steps        (R) Steps for redundant systems        x
x                                                                                x
x    REQUIRED                                                                    x
x    (E) Set default gateway pool       (V) Configure VLANs & networking       x
x    (H) Set host name                  (W) Configure web servers              x
x    (P) Set root password                                                      x
x                                                                                x
x    OPTIONAL                                                                    x
x    (D) Configure DNS                  (O) Configure remote access            x
x    (F) Configure FTP                  (S) Configure SSH                      x
x    (I) Initialize iControl portal     (T) Configure Telnetd                  x
x    (K) Set keyboard type              (U) Configure RSH                      x
x    (M) Define time servers            (Z) Set time zone                      x
x                                       (Q) Quit                               x
x                                                                                x
x    Enter Choice:                                                              x
x                                                                                x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
```

Figure 2.1 *The Setup utility menu. Some of these options may not be available on your system.*

Options available only through the Setup utility menu

This section contains descriptions of options that are available only through the Setup utility menu. These options include:

- Initializing the iControl portal
- Configuring RSH
- Configuring remote access

Initialize the iControl portal

This option is available in the menu only after you create the initial software configuration. Select this option to configure the CORBA ports (IIOP and FSSL). This option prompts you for a list of IP addresses or host names you want to embed as objects in the Portal object reference. Typically, in a redundant system, this list includes the fail-over IP address of the other BIG-IP in the redundant system.

This option prompts you to set the Portal to use IP addresses instead of DNS names. If the Portal is set to use IP addresses, the BIG-IP does not have to do a DNS lookup.

In addition to these settings, you can change the following iControl portal settings:

- You can set the security mode of the portal. You can allow the portal to handle non-secure requests.

- You can change the name of the Portal object reference file.
- You can specify the Portal PID file name.

Configuring RSH

This option is available only in the menu after you create the initial software configuration. Use this option to configure the remote shell (**rshd**) server. This utility prompts you for an IP address from which administrators may access the BIG-IP. You can use wildcard characters (*****) to include all addresses from a specific part of the network. This utility also prompts you to create a support account for access by technical support.

If **inetd** is not currently configured, this utility configures **inetd** for the remote shell server (**rshd**). If the service port for **rsh** is closed, this utility opens the service port to permit **rsh** connections to the system.

Configuring remote access

This option is available only in the menu after you create the initial software configuration. Use this options to modify remote access settings.

3

Additional Base Network Configuration

- Introduction
- Interfaces
- VLANs
- Self IP addresses
- Trunks
- Spanning Tree Protocol (STP)
- Port Mirroring

Introduction

Setting up the base network for BIG-IP means configuring elements such as the BIG-IP host name, a default gateway pool, interface media settings, and VLANs and self IP addresses. Configuration tasks for the BIG-IP base network are performed using the BIG-IP Setup utility. For information on using the Setup utility, see Chapter 2, *Using the Setup Utility*.

Once you have configured the base network elements with the Setup utility, you might want to further enhance the configuration of these elements. This chapter provides the information you need to perform these additional configuration tasks. You can perform these tasks using either the Configuration utility or the **bigpipe** command.

Elements you might want to further configure after running Setup are:

- **Interfaces**
  You can set the media type and the duplex mode for an interface, as well as display interface status.
- **VLANs**
  VLAN options include ***tagging*** and assigning interfaces to VLANs. In addition, you can group separate VLANs together for the purpose of bridging packets between them.
- **Self IP addresses**
  You can change self IP addresses or create any number of additional self IP addresses for a VLAN.

If your BIG-IP is an IP Application Switch, you also have three other BIG-IP features you can configure:

- **Trunks**
  Trunks are aggregated links. In link aggregation, interfaces can be combined into a trunk to increase bandwidth in an additive manner. The other benefit of link aggregation is link fail-over. If one link in a trunk goes down, traffic is simply redistributed over the remaining links.
- **Spanning Tree Protocol** (STP)
  STP domains provide for loop resolution in configurations where one or more external switches is connected in parallel with an IP Application Switch.
- **Port mirroring**
  This allows you to copy traffic from any interface or set of interfaces on a BIG-IP Application Switch to a single, separate interface. Typically you would install a sniffer device on the target port for debugging and/or monitoring.

Like interfaces, VLANs, and self IP addresses, these features can be configured using either the Configuration utility or the **bigpipe** command.

◆ Note

Once you have configured the base network, you can configure the high-level network. Examples of elements you configure as part of the high-level network are: Pools, rules, proxies, and network address translation (SNATs and NATs). For information on how to configure your high-level network, see Chapter 4, ***Configuring the High-Level Network****.*

Interfaces

A BIG-IP can have as few as two network interfaces and as many as twenty-nine. Before performing configuration tasks such as displaying interface status and settings, setting the media type, and setting the duplex mode, it is helpful to understand interface naming conventions.

Interface naming conventions

By convention, the Ethernet interfaces on a BIG-IP take the name **<s>.<p>** where **s** is the slot number of the NIC, and **p** is the port number on the NIC. As shown in Figure 3.1, for the 4U platform, slot numbering is left-to-right, and port numbering is top-to-bottom. Note that **slot 1** is reserved for the onboard NIC whether or not it is present.

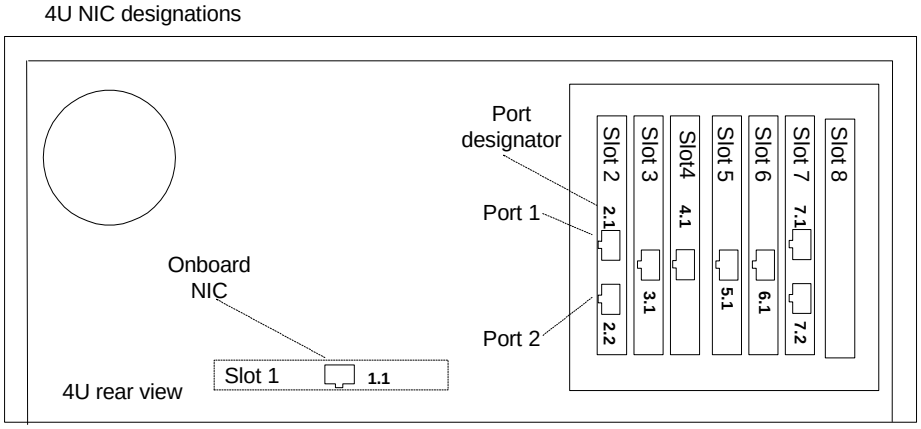

Figure 3.1 *Vertical slot and port numbering*

For the 2U platform, slot numbering is top-to-bottom and port numbering is left-to-right as shown in Figure 3.2.

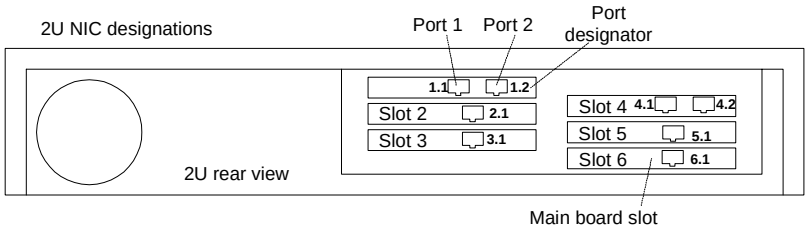

Figure 3.2 *Horizontal slot and port numbering*

For the Application Switch, slot numbering is left-to-right and port numbering is top-to-bottom as shown in Figure 3.3. Note that slot 2 is used for the gigabit ports, and slot 3 for a dedicated administrative port.

When a **bigpipe** command calls for a list of interfaces, the list may consist of one or more interfaces, with multiple interfaces separated by spaces. For example:

```
2.1 2.2 2.4 2.6
```

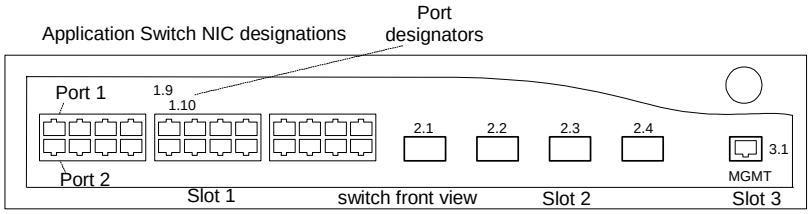

Figure 3.3 *Application Switch slot and port numbering*

Displaying status and settings for interfaces

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

```
b interface show
```

Figure 3.4 is an example of the output you see when you issue this command on an active/standby unit in active mode.

```
interface  speed   pkts   pkts   pkts   pkts   bits   bits errors trunk STP
           Mb/s     in    out   drop   coll     in    out
  5.1 UP  100 HD     0    213      0      0      0  74.2K      0
  4.1 UP  100 HD    20     25      0      0  28.6K  33.9K      0
```

Figure 3.4 *The **bigpipe interface show** command output*

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

```
b interface <if_name> show
```

Media type and duplex mode

Properties that are configurable on the interfaces include media type and duplex mode, as shown in Table 3.1.

Interface Properties	Description
media	You may specify a media type or use **auto** for automatic detection.
duplex	You may specify a full or half duplex mode, or use **auto** for automatic selection.

Table 3.1 *The attributes you can configure for an interface*

Setting the media type

You can set the media type to the specific media type for the interface card or to **auto** for auto detection. If the media type is set to **auto** and the card does not support auto detection, the default type for that interface is used, for example **1000BaseTX**.

Use the following syntax to set the media type:

```
b interface <if_name> media <media_type> | auto
```

(Default media type is **auto**.)

◆ Note

If the BIG-IP is inter-operating with an external switch, the media setting should match that of the switch. For more information, see ***Setting the interface media type****, on page 2-6.*

Setting the duplex mode

You can set duplex mode to full or half duplex. If the media type does not allow duplex mode to be set, this is indicated by an onscreen message. If media type is set to **auto**, or if setting duplex mode is not supported for the interface, the duplex setting is not saved to **bigip_base.conf**.

Use the following syntax to set the duplex mode:

```
b interface <if_name> duplex  full | half | auto
```

(Default mode is **auto**.)

VLANs

A ***VLAN*** is a grouping of separate networks that allows those networks to behave as if they were a single local area network, whether or not there is a direct ethernet connection between them.

BIG-IP offers several options that you can configure for a VLAN. These options are summarized in Table 3.2.

Option	Description
Create a default VLAN configuration	Use the Setup utility to create a default VLAN configuration.
Create, rename, or delete VLANs	Create, rename, or delete a VLAN.
Configure packet access to VLANs	Through an option called ***tagging***, you can direct packets from multiple VLANs to a specific BIG-IP interface, or direct traffic from a single VLAN to multiple interfaces.
Manage the L2 forwarding table	You can edit the L2 forwarding table to enter static MAC address assignments.
Create VLAN groups	Create a VLAN group to allow layer 2 packet forwarding between VLANs.
Set VLAN security	You can set port lockdown by VLAN.
Set fail-safe timeouts	You can set a failsafe timeout on a VLAN. You can use a failsafe timeout to trigger fail-over in a redundant system.
Set self IP addresses	You can set one or more self IP addresses for VLANs.
Set MAC masquerade	You can use the MAC masquerade to set up a media access control (MAC) address that is shared by a redundant system.

Table 3.2 *Configuration options for VLANs*

Default VLAN configuration

By default, the Setup utility configures each interface on the BIG-IP as a member of a VLAN. The BIG-IP identifies the fastest interfaces, makes the lowest-numbered interface in that group a member of the VLAN **external,** and makes all remaining interfaces members of the VLAN **internal**. This creates the mapping shown in Figure 3.5.

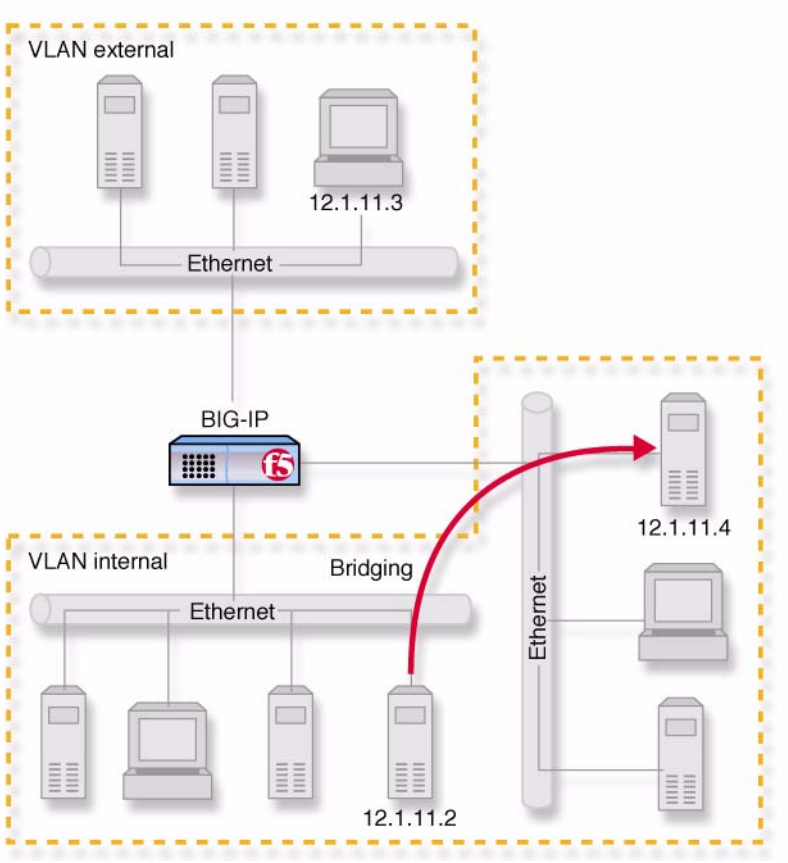

Figure 3.5 *Default VLAN configuration*

As Figure 3.5 shows, VLAN flexibility is such that separate IP networks can belong to a single VLAN, while a single IP network can be split among multiple VLANs. (The latter case allows the BIG-IP to be inserted into an existing LAN without renaming the nodes.) The VLANs named **external** and **internal** are separate networks, and in the configuration shown they behave like separate networks. The networks belonging to VLAN **internal** are also separate networks, but have been made to behave like a single network. This is accomplished using a feature called VLAN ***bridging***.

Your default VLAN configuration is created using the Setup utility. On a typical unit with two interfaces, you create an internal and external VLAN.

Creating, renaming, and deleting VLANs

Typically, if you use the default configuration, one VLAN is assigned to each interface. However, if you need to change your network configuration, or if the default VLANs are not adequate for a network configuration, you can create new VLANs, rename existing VLANs, or delete a VLAN.

To create a VLAN using the Configuration utility

1. In the navigation pane, click **Network**.
   The VLANs screen opens.
2. Click the **Add** button.
3. Type the attributes for the VLAN.
4. Click **Done**.

To rename or delete a VLAN using the Configuration utility

1. In the navigation pane, click **Network**.
   The VLANs screen opens.
2. In the VLANs screen, use one of the following options:
   - To rename a VLAN, click the VLAN name you want to change. The VLAN properties screen opens. Type the new name in the **VLAN name** box.
   - To delete a VLAN, click the **Delete** button for the VLAN you want to delete.
3. Click **Done**.

To create, rename, or delete a VLAN from the command line

To create a VLAN from the command line, use the following syntax:

```
b vlan <vlan name> interfaces add <if name> <if name>
```

For example, if you want to create a VLAN named **myvlan** that contains the interfaces **1.1** and **1.2**, type the following command:

```
b vlan myvlan interfaces add 1.1 1.2
```

To rename an existing VLAN, use the following syntax:

```
b vlan <vlan name> rename <new vlan name>
```

For example, if you want to rename the VLAN **myvlan** to **yourvlan**, type the following command:

```
b vlan myvlan rename yourvlan
```

To delete a VLAN, use the following syntax:

```
b vlan <vlan name> delete
```

For example, to delete the VLAN named **yourvlan**, type the following command:

```
b vlan yourvlan delete
```

Configuring packet access to VLANs

The BIG-IP supports two methods for sending and receiving packets through an interface that is a member of one or more VLANs. These two methods are:

- **Port-based access to VLANs** - Packets are accepted for a VLAN because the packets have no tags in their headers and were received on an interface that is a member of a VLAN. With this method, an interface is configured as an ***untagged*** member of the VLAN. Packets sent out through untagged interfaces contain no tag in their header.
- **Tag-based access to VLANs** - Packets are accepted for a VLAN because the packets have tags in their headers and the tag matches the VLAN identification number for the VLAN. With this method, an interface is configured as a ***tagged*** member of the VLAN. Packets sent out through tagged interfaces contain a tag in their header.

The method used by a VLAN is determined by the way that you add a member interface to a VLAN. When creating a VLAN or modifying VLAN properties (using the Configuration utility or the **bigpipe** command), you can add an interface to that VLAN as either an ***untagged*** or a ***tagged*** interface.

The following two sections describe these two methods of providing packet access to a VLAN.

Port-based access to VLANs

Port-based access to VLANs occurs when an interface is added to a VLAN as an ***untagged*** interface. In this case, the interface can be added only to that VLAN and to no others. This limits the interface to accepting traffic only from that VLAN, instead of from multiple VLANs. To solve this problem, BIG-IP allows you to configure a feature known as ***tagging***, described in the following section.

Tag-based access to VLANs

Tag-based access to VLANs occurs when an interface is added to a VLAN as a ***tagged*** interface. A tagged interface can be added to multiple VLANs, thereby allowing the interface to accept traffic from each VLAN of which the interface is a member.

When you add an interface to a VLAN as a tagged interface, BIG-IP associates the interface with the VLAN identification number, or ***tag***, which becomes embedded in a header of a packet.

◆ Note

Every VLAN has a VLAN identification number. This identification number is assigned to a VLAN either explicitly by a user, when creating the VLAN, or automatically by BIG-IP, if the user does not supply one.

Each time you add an interface to a VLAN, either when creating a VLAN or modifying its properties, you can designate that interface as a ***tagged*** interface. A single interface can therefore have multiple tags associated with it.

The result is that whenever a packet comes into that interface, the interface reads the tag that is embedded in a header of the packet. If the tag in the packet matches any of the tags associated with the interface, the interface accepts the packet. If the tag in the packet does *not* match any of the tags associated with the interface, the interface rejects the packet.

Example

Figure 3.6 shows the difference between using three untagged interfaces (where each interface must belong to a separate VLAN) versus one tagged interface (which belongs to multiple VLANs).

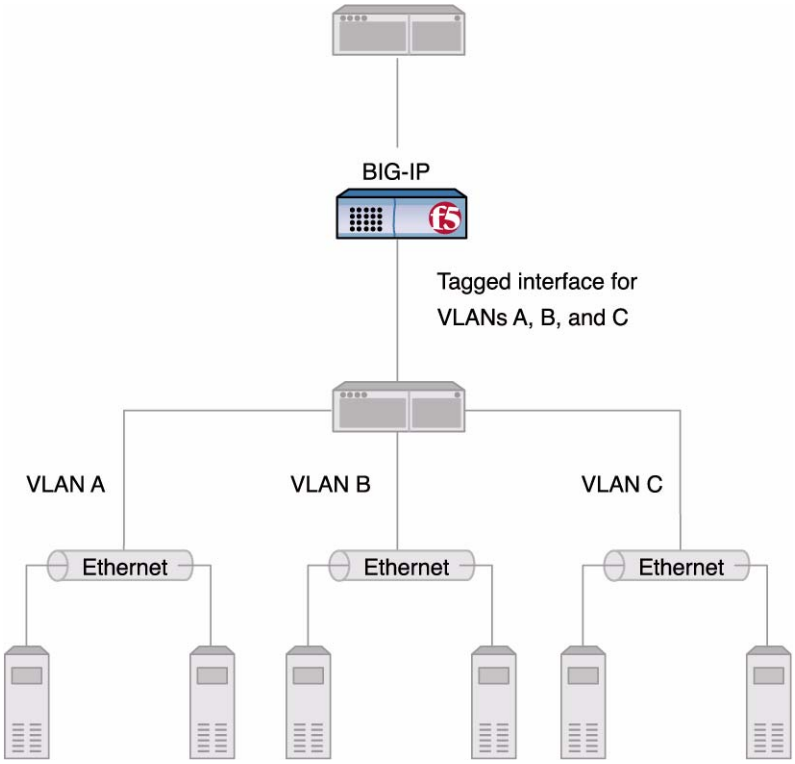

Figure 3.6 *Equivalent solutions using untagged and tagged interfaces*

The configuration on the left shows a BIG-IP unit with three internal interfaces, each a separate, untagged interface. This is a typical solution for supporting three separate customer sites. In this scenario, each interface can only accept traffic from its own VLAN.

Conversely, the configuration on the right shows a BIG-IP with one internal interface and an external switch. The switch places the internal interface on three separate VLANs. The interface is configured on each VLAN as a tagged interface. In this way, the single interface becomes a tagged member of all three VLANs, and accepts traffic from all three. The configuration on the right is the functional equivalent of the configuration on the left.

Not only can you add a single, tagged interface to multiple VLANs, as shown in the above example, you can also add multiple tagged interfaces to a single VLAN.

Configuration procedures

You configure tag-based access to VLANs using either the Configuration utility or the **bigpipe vlan** command. You can configure tag-based access either when you create a VLAN and add member interfaces to it, or by modifying the properties of an existing VLAN. In the latter case, you simply change the status of one or more member interfaces from untagged to tagged.

To create a VLAN that supports tag-based access using the Configuration utility

Creating a VLAN that supports tag-based access means creating the VLAN and then adding one or more tagged interfaces to it.

1. In the navigation pane, click **Network**.
   The VLAN screen opens.
2. Click the **Add** button.
   The Add VLAN screen opens.
3. On the Add VLAN screen, type the VLAN name.
4. In the VLAN tag box, you can optionally specify a VLAN ID number. If you do not provide one, BIG-IP assigns a default number.
5. In the **Resources** box, specify any tagged interfaces by selecting the appropriate interface numbers from the **Interface Number** list and clicking **tagged** >>.
6. Configure the other VLAN options.
7. Click **Done**.

To configure tag-based access on an existing VLAN using the Configuration utility

Configuring tag-based access on an existing VLAN means changing the existing status of one or more member interfaces from **untagged** to **tagged**.

1. In the navigation pane, click **Network**.
   The VLAN screen opens.
2. Click the VLAN name in the list.
   The properties screen for that VLAN opens.
3. In the **Resources** box, move any untagged interfaces from the **Current Interfaces** list to the **Interface Number** list.
4. Specify any tagged interfaces by selecting the appropriate interface numbers from the **Interface Number** list and clicking **tagged** >>**.**
5. Click **Done**.

To create a VLAN that supports tag-based access from the command line

1. Type the **bigpipe vlan** command, specifying a VLAN name, the **tag** keyword, and a VLAN ID number. The following example creates the VLAN **external** with a VLAN ID of **1209**.

```
b vlan external tag 1209
```

2. Add the interfaces to the VLAN **external** as tagged interfaces. This is done by specifying the VLAN name, the **tagged** keyword, and the interfaces to be tagged. For example:

```
b vlan external interfaces add tagged 4.1 5.1 5.2
```

The effect of this command is to associate a tag with interfaces **4.1**.and **5.1,** which in turn allows packets with that tag access to the **external** VLAN.

The above procedure adds multiple tagged interfaces to a single VLAN. However, you can also add a single tagged interface to multiple VLANs (similar to the scenario presented in Figure 3.6). This results in a single interface having more than one tag associated with it. For example, the following commands add the tagged interface 4.1 to the two VLANs **external** and **internal**:

```
b vlan external interfaces add tagged 4.1
b vlan internal interfaces add tagged 4.1
```

Managing the Layer 2 forwarding table

Layer 2 forwarding is the means by which packets are exchanged directly between nodes on separate VLANs that are members of the same VLAN group, as described in *Creating VLAN groups*, on page 3-14. This is accomplished using a simple forwarding table for each VLAN with **proxy forward** enabled. The forwarding table has an entry for each node in the VLAN and associates the MAC address of that node with the BIG-IP interface using the following format:

```
<MAC address> -> <if>
```

For example:

```
00:a0:c9:9e:1e:2f -> 4.1
```

Viewing and editing the L2 forwarding table

You can view the L2 forwarding table, delete entries, and add static entries. The entries that appear in the table automatically are learned and periodically updated and are called ***dynamic entries***. Entries that you add to the table manually are called ***static entries***. Static entries are not automatically updated. Entering static entries is useful if you have network devices that do not advertise their MAC addresses.

You can view and edit the L2 forwarding table using the **bigpipe vlan <vlan_name> fdb** command. The **<vlan_name>** may be either a VLAN or a VLAN group.

To view the L2 forwarding table from the command line

Type the following command:

```
b vlan <vlan name> fdb show
```

For example:

```
b vlan internal fdb show
```

This produces a display such as the following:

```
Forwarding table --
   00:40:05:30:cc:94 -> 5.1)
```

To view L2 forwarding table static entries from the command line

Type the following command:

```
b vlan <vlan name> fdb show static
```

For example:

```
b vlan internal fdb show static
```

To view L2 forwarding table dynamic entries from the command line

Type the following command:

```
b vlan <vlan name> fdb show dynamic
```

For example:

```
b vlan internal fdb show dynamic
```

To add an entry to the L2 forwarding table from the command line

Type the following command:

```
b vlan <vlan name> fdb add <MAC address> interface <ifname>
```

For example:

```
b vlan internal fdb add <MAC address> interface <ifname>
```

To delete an entry from the L2 forwarding table from the command line

Type the following command:

```
b vlan <vlan name> fdb delete <MAC address> interface <ifname>
```

For example:

```
b vlan <vlan name> fdb delete 00:a0:c9:9e:1e:2f interface 4.1
vlan <vlan name> fdb show static
vlan <vlan name> fdb show dynamic
vlan <vlan name> fdb show
```

Setting the L2 forwarding aging time

Entries in the L2 forwarding table have a specified life span, after which they are flushed out if the MAC address is no longer present on the network. This process is called the ***L2 forward aging time*** and you can set it using the global variable **L2 Aging Time**. The default value is 300 seconds.

To set the L2 forwarding aging time using the Configuration utility

1. In the navigation pane, click **System**.
   The System Properties screen opens.
2. Click the **Advanced Properties** tab.
3. In **L2 Aging Time** box, enter the aging time in seconds.
4. Click **Done**.

To set the L2 forwarding aging time from the command line

Type the following command:

```
b global l2_aging_time <time_in_seconds>
```

For example:

```
b global l2_aging_time 200
```

Creating VLAN groups

A ***VLAN group*** is a grouping of two or more VLANs belonging to the same IP network for the purpose of allowing layer 2 packet forwarding, also known as L2 forwarding, between those VLANs. L2 forwarding is the equivalent of bridging where you want communication between VLANs. By creating a VLAN group, nodes on the separate VLANs can exchange packets directly.

In the example shown in figure 3.5, VLANs **external** and **internal** represent separate networks that were originally a single network. You can make them behave like a single network again much like the networks contained in VLAN **internal**. You accomplish this by grouping them as shown in Figure 3.7.

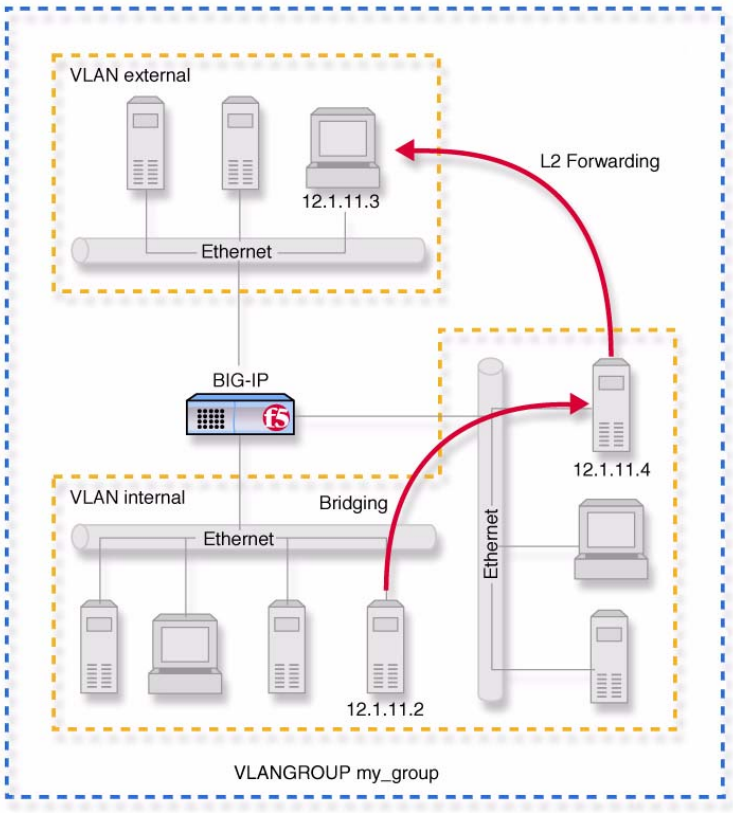

Figure 3.7 *VLANs and a VLAN group*

To configure a VLAN group to use layer 2 forwarding, you must:

- Create the VLAN group.
- Assign a self IP address to the VLAN group, for routing purposes.
- Verify that layer 2 forwarding (also known as **proxy forwarding**) is enabled.

The following sections describe these procedures.

To create a VLAN group

You can create a VLAN group from the command line using the **vlangroup** command. For example:

```
b vlangroup network11 vlans add internal external
```

To assign the self IP address to the VLAN group

You can assign a self IP address to the VLAN group using the **bigpipe** command, as follows:

```
b self <ip address> vlan <vlangroup name>
```

To verify that Layer 2 forwarding is enabled

Layer 2 forwarding is enabled for the VLAN group using the **vlan proxy_forward** attribute. This attribute is enabled by default when the VLAN group is enabled. To verify that proxy forwarding is enabled, type the following command:

```
b vlans show
```

Setting up security for VLANs

You can lock down a VLAN to prevent direct connection to the BIG-IP through that VLAN. You can override this lockdown for specific services by enabling the corresponding global variable for that service. For example:

```
b global open_ssh_port enable
```

To enable or disable port lockdown using the Configuration utility

1. In the navigation pane, click **Network**.
   The VLAN screen opens.
2. Click the VLAN name in the list.
   The properties screen for that VLAN opens.
3. To enable port lockdown, click a check in the **Port Lockdown** box.
   To disable port lockdown, clear the **Port Lockdown** check box.
4. Click **Done**.

To enable or disable port lockdown from the command line

To enable port lockdown, type:

```
b vlan <vlan_name> port_lockdown enable
```

To disable port lockdown, type:

```
b vlan <vlan_name> port_lockdown disable
```

Setting fail-safe timeouts for VLANs

For redundant BIG-IP pairs, you can enable a failsafe mechanism that will fail over when loss of traffic is detected on a VLAN, and traffic is not restored during the fail-over timeout period for that VLAN. You can enable a fail-safe mechanism to attempt to generate traffic when half the timeout has elapsed. If the attempt is successful, the fail-over is stopped.

To set the fail-over timeout and arm the fail-safe using the Configuration utility

1. In the navigation pane, click **Network**.
   The VLAN screen opens.
2. Click the VLAN name in the list.
   The properties screen for that VLAN opens.
3. Check the **Arm Failsafe** box and specify the timeout in seconds in the **Timeou**t box.

To set the fail-over timeout and arm the fail-safe from the command line

Using the **vlan** command, you may set the timeout period and also arm or disarm the fail-safe.

To set the timeout, type:

```
b vlan <vlan_name> timeout <timeout_in_seconds>
```

To arm the fail-safe, type:

```
b vlan <vlan_name> failsafe arm
```

To disarm the fail-safe, type:

```
b vlan <vlan_name> failsafe disarm
```

Setting the MAC masquerade address

You can share the media access control (MAC) masquerade address between BIG-IP units in a redundant pair. This has the following advantages:

- Increased reliability and failover speed, especially in lossy networks
- Interoperability with switches that are slow to respond to the network changes
- Interoperability with switches that are configured to ignore network changes

The MAC address for a VLAN is the MAC address of the first interface to be mapped to the VLAN, typically 4.1 for external and 5.1 for internal. You can view the interfaces mapped to a VLAN using the following command:

```
b vlan show
```

You can view the MAC addresses for the interfaces on the BIG-IP using the following command:

```
b interface show verbose
```

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

```
b vlan <vlan_name> mac_masq <MAC_addr>
```

Find the MAC address on both the active and standby units, and pick one that is similar but unique. A safe technique for selecting the shared MAC address follows.

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

```
Active: 3.1 = 0:0:0:ac:4c:a2
Standby: 3.1 = 0:0:0:ad:4d:f3
```

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

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

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

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

◆ Note

*The MAC masquerade information is stored in the **bigip_base.conf** file.*

Self IP addresses

A ***self IP address*** is an IP address mapping to one or more VLANs and their associated interfaces on a BIG-IP. You assign a self IP address to each interface on the unit as part of Setup configuration, and you also assign a floating (shared) alias for units in a redundant pair. (A floating self IP address is the address to which the servers behind the BIG-IP route traffic). You can create additional self IP addresses for health checking, gateway failsafe, routing, or other purposes. You can create these additional self IP addresses using the **self** command.

To add a self IP address to a VLAN using the Configuration utility

1. In the navigation pane, click **Network**.
   The VLANs screen opens.
2. Click the Self IP Addresses tab.
3. Click the **Add** button.

4. In the **IP Address** box, type the self IP address to be assigned.
5. In the **Netmask** box, type an optional netmask.
6. In the **Broadcast** box, type an optional broadcast address.
7. If you want to configure the self IP address as a floating address, check the **Floating** box.
8. If you want to enable the address for SNAT auto-mapping, check the **SNAT Automap** box.
9. In the **VLAN** box, type the name of the VLAN to which you want to assign the self IP address.
10. Click **Done**.

To add a self IP address to a VLAN from the command line

Use the following syntax:

```
b self <addr> vlan <vlan_name> [ netmask <ip_mask> ][ broadcast <broadcast_addr>] [unit
    <id>]
```

You can add any number of additional self IP addresses to a VLAN to create aliases. For example:

```
b self 11.11.11.4 vlan external
b self 11.11.11.5 vlan external
b self 11.11.11.6 vlan external
b self 11.11.11.7 vlan external
```

Also, any one self IP address may have **floating** enabled to create a ***floating alias*** that is shared by both units of a BIG-IP redundant pair:

```
b self 11.11.11.8 floating enable
```

Assigning a self IP address to an interface automatically maps it to the VLAN of which it is a member. Assigning a self IP address to an interface not mapped to an untagged VLAN produces an error message.

Enabling or disabling SNAT automap

The self IP addresses you enable on the external VLAN determine the translation address for SNAT auto-mapping. For more information about SNAT auto-mapping, refer to *Configuring SNAT automapping*, on page 4-126.

Trunks

Link aggregation is the grouping of links (individual physical interfaces) to form a ***trunk***. Link aggregation increases the bandwidth of the individual links in an additive manner. Thus, four fast Ethernet links, if aggregated,

create a single 400 Mbps link. The other advantage of link aggregation is link fail-over. If one link in a trunk goes down, traffic is simply redistributed over the remaining links.

A trunk must have a controlling link, and acquires all the attributes of that controlling link from layer 2 and above. The trunk automatically acquires the VLAN membership of the controlling link but does not acquire its media type and speed. Outbound packets to the controlling link are load balanced across all of the known-good links in the trunk. Inbound packets from any link in the trunk are treated as if they came from the controlling link.

A maximum of eight links may be aggregated. For optimal performance, links should be aggregated in powers of two. Thus, you ideally will aggregate two, four, or eight links.

To configure a trunk using the Configuration utility

1. In the navigation pane, click **Network**.
   The Network screen opens.
2. Click the **Trunks** tab.
   The Trunks screen opens.
3. Click the **Add** button.
4. Select the link that is to be the controlling link from the Available Interfaces list, and click **controlling** >>.
   The interface appears at the top of the Aggregated Interfaces list.
5. Select the remaining link(s) from the Available Interfaces list and click **aggregated** >>.
   The interface(s) appears in the Aggregated Interfaces list below the controlling link.
6. Click **Done**.

To configure a trunk from the command line

Use the following syntax to configure a trunk from the command line:

```
b trunk <controlling_if> define <if_list>
```

Interfaces are specified using the **s.p** convention, where **s** is slot number and **p** is port number. An **<if_list>** is one or more such interfaces, with multiple interfaces separated by spaces.

For more information on interface naming, refer to *Interface naming conventions*, on page 3-2.

Spanning Tree Protocol (STP)

The BIG-IP Application Switch provides Spanning Tree Protocol (STP) implementation for loop resolution in configurations where one or more external switches is connected in parallel with the BIG-IP. You can use this

feature to configure two or more interfaces on the unit as an STP domain. For interfaces in the STP domain, the spanning tree algorithm identifies the most efficient path between the network segments, and establishes the switch associated with that path as the **root**. Links forming redundant paths are shut down, to be re-activated only if the root fails.

The STP domain should contain all ports that are connected in parallel to an external switch where there are nodes on the link capable of generating or receiving traffic. A second domain is called for if there is an additional switch or switches connected in parallel with additional BIG-IP interfaces.

Use of STP may slow performance significantly, particularly if more than one STP domain is created, and may have unforeseen effects on complex networks. It is important to test your STP configuration before placing it online.

Creating and deleting STP domains

You can create or delete STP domains using the Configuration utility or from the command line.

To create an STP domain using the Configuration utility

1. In the navigation pane, click **Network**.
   The Network screen opens.
2. Click the **STP** tab.
   The Trunks screen opens.
3. Click the **Add** button.
4. Configure the STP domain attributes.
5. Click **Done**.

To create or delete an STP domain from the command line

To create an STP domain from the command line, use the following syntax:

```
b stp <stp_name> interfaces add <if _list> | all
```

For example, if you want to create an STP domain named **mystp** that contains the interfaces 1.1 and 1.2, type the following command.

```
b stp mystp interfaces add 1.1 1.2
```

If you want to create an STP domain named **mystp** that contains all interfaces on the BIG-IP, type:

```
b stp <stp_name> interfaces add all
```

To delete an STP domain, use the following syntax:

```
b stp <stp_name> delete
```

Setting time intervals for an STP domain

You can set the time intervals in seconds for **hello**, **max_age**, and **forward_delay** for the STP domain from the command line using the following syntax:

```
b stp <stp_name> hello <interval>
b stp <stp_name> max_age <interval>
b stp <stp_name> forward_delay <interval>
```

Adding or deleting interfaces in an STP domain

To add interfaces to an STP domain from the command line, use the following syntax:

```
b stp <stp_name> interfaces add <if _list>
```

To delete interfaces from an STP domain, use the following syntax.

```
b stp <stp_name> interfaces delete <if _list>
```

Disabling and re-enabling an STP domain

To disable an STP domain from the command line, use the following syntax:

```
b stp <stp_name> disable
```

To re-enable interfaces on an STP domain, use the following syntax:

```
b stp <stp_name> enable
```

◆ Note

Disabling or deleting all interfaces on an STP domain disables the domain. You cannot re-enable the domain without adding interfaces.

Disabling and re-enabling interfaces in an STP domain

To disable specific interfaces in the STP domain from the command line, use the following syntax:

```
b stp <stp_name> interfaces disable <if_list>
```

To re-enable interfaces in an STP domain, use the following syntax:

```
b stp <stp_name> interfaces enable <if_list>
```

Restarting stpd

The **stpd** does not automatically restart when you synchronize configurations between units in a BIG-IP redundant pair. In order to restart the **stpd**, type the following command:

```
bigstart restart stpd
```

Port Mirroring

For the IP Application Switch, you can copy traffic from any port or set of ports to a single, separate port. This is called ***port mirroring***. You should attach a sniffer device to the target port (called the ***mirror-to*** port) for debugging and/or monitoring.

Setting up a port mirror

Port mirroring consists of specifying a mirror-to port and adding to it one or more ports (that is, a port list) to be mirrored. You can set up port mirroring using the Configuration utility or from the command line.

To set up port mirroring using the Configuration utility

1. In the navigation pane, click **Network**.
   The Network screen opens.
2. Click the **Interfaces** tab.
3. Click the **Port Mirroring** subtab.
4. In the Port Mirroring screen, configure the port mirror attributes.
5. Click **Done**.

To set up port mirroring from the command line

Use this **bigpipe** syntax for setting up port mirroring:

```
b mirror <mirror_to_if> interfaces add <if_list>
```

Example:

```
b mirror 3.24 interfaces add 3.1 3.3 3.10
```

Deleting interfaces from a port mirror or deleting a port mirror

You can delete individual interfaces from a port mirror, or you can completely delete a port mirror.

To delete interfaces from the port mirror using the command line

Use this **bigpipe** syntax to delete interfaces from the port mirror:

```
b mirror <mirror_to_if> interfaces delete <if_list>
```

For example:

```
b mirror 3.24 interfaces delete 3.10
```

To delete the port mirror from the command line

Use this **bigpipe** syntax to delete the port mirror:

```
b mirror <mirror_to_if> delete
```

For example:

```
b mirror 3.24 delete
```

4

Configuring the High-Level Network

- Introduction
- Pools
- Rules
- Virtual servers
- Proxies
- Nodes
- Services
- Address translation: SNATs, NATs, and IP forwarding
- Health monitors

Introduction

This chapter describes the elements that make up the ***high-level network*** of BIG-IP. The high-level network is distinct from the ***base network***, which is configured with the Setup utility.

Just as the base network is built on the BIG-IP interfaces, the high-level network is built on the load balancing ***pool***. The high-level network includes all of the properties associated with pools, as well as virtual servers, and nodes. It can also include pool-selection rules, as well as services, proxies, SNATs, NATs, and health monitor associations for nodes.

- *Pools* represent groups of nodes that can receive traffic from BIG-IP according to a specified load balancing method.
- *Rules* enable a virtual server to choose among multiple pools based on selection criteria. In the form of ***cache rules,*** they also allow the virtual server to cache content intelligently based on frequency of access.
- *Proxies* are used for SSL acceleration and content conversion (akamaization) where these features are present.
- *Virtual Servers* can be of four types: standard, wildcard, network, or forwarding.
- *Proxies* are used for SSL acceleration and content conversion (akamaization) where these features are present.
- *Services* correspond to the ports (for example, port **80** and port **443**) specified for nodes as they are defined in load balancing pools. Service options include enabling/disabling of service, connection limits, and timeouts for UDP and TCP.
- *SNATs* and *NATs* are secure network address translations and network address translations, respectively, and are used primarily to allow servers to establish outgoing connections as clients.
- *Health monitors* are status checking devices that may be configured by the user, and are associated with nodes for ongoing monitoring.

The remaining sections of this chapter describe each of these elements and the procedures for configuring them for BIG-IP.

Pools

A load balancing pool is the primary object in the high-level network. A ***pool*** is a set of devices grouped together to receive traffic according to a load balancing method. When you create a pool, the members of the pool become visible nodes on the high-level network and can acquire the various properties that attach to nodes. Pools can be accessed through a virtual server, either directly or through a rule, which chooses among two or more pools. The Rules section of this chapter describes several ways to select pools using rules.

You can use the Configuration utility or the **bigpipe pool** command to create, delete, modify, or display the pool definitions on the BIG-IP.

When creating a pool, you can configure various pool attributes. Table 4.1 lists the attributes you can configure for a pool.

Pool Attributes	Description	Required or Optional?
Pool name	You can define the name of the pool.	Required
Member specification	You can define each network device, or node, that is a member of the pool.	Required for non-forwarding pools
Load balancing method	You can define a specific load balancing mode for a pool, and you can configure priority-based member activation. Various pools can be configured with different load balancing modes.	Required (Default=Round Robin)
Persistence method	You can define a specific persistence method for a pool. You can have various pools configured with different persistence methods.	Optional
HTTP redirection	You can redirect HTTP requests to a fallback host, protocol, port, or URI path.	Optional
HTTP header insertion	You can configure a pool to insert a header into an HTTP request. For example, the header could include an original client IP address, to preserve the address during a SNAT connection.	Optional
Quality of Service (QoS) level	You can configure a pool to set a specific QoS level within a packet, based on the targeted pool.	Optional
Type of Service (ToS) level	You can configure a pool to set a specific ToS level within a packet, based on the targeted pool.	Optional
Disabling of SNAT and NAT connections	You can configure a pool so that SNATs and NATs are automatically disabled for any connections using that pool.	Optional
Forwarding	You can configure a forwarding pool, which causes a connection to be forwarded, using IP routing, instead of load balanced. Creating a forwarding pool allows you to use pool-based features for traffic that should be forwarded.	Optional

Table 4.1 *The attributes of a pool*

Working with pools

You can manage pools using either the web-based Configuration utility or the command-line interface. This section describes how to create, delete, modify, or display a pool, using each of these configuration methods.

To create a pool using the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. Click the **Add** button.
   The Add Pool screen opens.
3. In the Add Pool screen, fill in the fields to create the new pool and configure its attributes.
4. Click **Done**.

To create a pool from the command line

To define a pool and configure its attributes from the command line, use the following syntax:

```
b pool <pool_name> { member <member_definition> ... member <member_definition> }
```

For example, if you want to create the pool **my_pool** with two members, you would type the following command:

```
b pool my_pool { member 11.12.1.101:80 member 11.12.1.100:80 }
```

Use the elements shown in Table 4.2 to construct pools from the command line. These elements correspond to the pool attributes listed in Table 4.1.

Pool Element	Syntax
Pool name	A string from 1 to 31 characters, for example: **new_pool**
Member definition	member <ip_address>:<service> [ratio <value>] [priority <value>]
lb_method_specificaton	lb_method [rr | ratio | fastest | least_conn | predictive | observed | ratio_member | fastest_member | least_conn_member | observed_member | predictive_member | dynamic_ratio]
persist_mode_specification	persist_mode [simple | cookie | ssl | sip | sticky | msrdp]
fallback_host_specification	fallback <fallback_host>
fallback_protocol_specification	fallback <fallback_protocol>
fallback_port_specification	fallback <fallback_port>
fallback_path_specification	fallback <fallback_path>
header insert	header_insert <quoted string>

Table 4.2 *Elements for pool construction*

Pool Element	Syntax
link_qos to client level	link_qos to client <level>
link_qos to server level	link_qos to server <level>
ip_tos to client level	ip_tos to client <level>
ip_tos to server level	ip_tos to server <level>
snat disable	snat <ip address> disable
nat disable	nat <ip address> disable
forward	forward

Table 4.2 *Elements for pool construction*

To delete a pool from the command line

To delete a pool, use the following syntax:

```
b pool <pool_name> delete
```

You must remove all references to a pool before you can delete a pool.

To modify a pool from the command line

In addition to adding nodes to a pool or deleting nodes from a pool, you can also modify pool attributes. You can add a new member to a pool, change the load-balancing mode, or delete a member from a pool.

For example, to change the default load-balancing mode from Round Robin to Predictive and add two new members to the pool, use a command such as the following:

```
b pool <pool_name> { \
lb_method predictive \
member 11.12.1.101:80 \
member 11.12.1.100:80 }
```

To display one or more pools from the command line

Use the following syntax to display all pools:

```
b pool show
```

Use the following syntax to display a specific pool:

```
b pool <pool_name> show
```

The following sections describe the various pool attributes that you can configure for a pool.

Pool Name

The most basic attribute you can configure for a pool is the pool name. Pool names are case-sensitive and may contain letters, numbers, and underscores (_) only. Reserved keywords are not allowed.

Each pool that you define must have a unique name.

Member specification

For each pool that you create, you must specify the nodes that are to be members of that pool. Nodes must be specified by their IP addresses.

Load balancing method

Load balancing is an integral part of the BIG-IP. Configuring load balancing on the BIG-IP means determining your load balancing scenario, that is, which node should receive a connection hosted by a particular virtual server. Once you have decided on a load balancing scenario, you can specify the appropriate load balancing method for that scenario.

The load balancing method is a pool attribute and consists of two properties: the load-balancing mode and priority-based activation.

Load balancing modes

A ***load balancing mode*** is an algorithm or formula that the BIG-IP uses to deterimine the node to which traffic will be sent. Individual load balancing modes take into account one or more dynamic factors, such as current connection count. Because each application of the BIG-IP is unique, and node performance depends on a number of different factors, we recommend that you experiment with different load balancing modes, and select the one that offers the best performance in your particular environment.

The default load balancing mode on the BIG-IP is ***Round Robin***, which simply passes each new connection request to the next server in line. All other load balancing modes take server capacity and/or status into consideration.

If the equipment that you are load balancing is roughly equal in processing speed and memory, Round Robin mode works well in most configurations. If you want to use the Round Robin mode, you can skip the remainder of this section, and begin configuring other pool attributes that you want to add to the basic pool configuration.

If you are working with servers that differ significantly in processing speed and memory, you may want to switch to Ratio mode or to one of the dynamic modes.

The individual load balancing modes are as follows.

Round Robin

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

Ratio

BIG-IP distributes connections among machines according to ratio weights that you define, where the number of connections that each machine receives over time is proportionate to a ratio weight you define for each machine. This is a static load balancing mode, basing distribution on static user-assigned ratio weights that are proportional to the capacity of the servers.

Load balancing calculations may be localized to each pool (member-based calculation) or they may apply to all pools of which a server is a member (node-based calculation). Member-based calculation is specified by the extension ***ratio_member.*** This distinction is especially important; in Ratio Member mode, the actual ratio weight is a member attribute in the pool definition, whereas in Ratio mode, the ratio weight is an attribute of the node.

Dynamic ratio

Dynamic Ratio mode is like Ratio mode except that ratio weights are based on continuous monitoring of the servers and are therefore continually changing.

This is a dynamic load balancing mode, distributing connections based on various aspects of real-time server performance analysis, such as the current number of connections per node or the fastest node response time.

Dynamic Ratio mode is used specifically for configuring RealNetworks RealServer platforms, Windows platforms equipped with Windows Management Instrumentation (WMI), or any server equipped with an SNMP agent such as the UC Davis SNMP agent or Windows 2000 Server SNMP agent. To install and configure the necessary server software for these systems, refer to *Configuring servers and the BIG-IP for Dynamic Ratio load balancing*, on page 4-10.

Fastest

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

Load balancing calculations may be localized to each pool (member-based calculation) or they may apply to all pools of which a server is a member (node-based calculation). The variant of the mode using member-based calculation is distinguished by the extension ***fastest_member***.

Least Connections

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

This is a dynamic load balancing mode, distributing connections based on various aspects of real-time server performance analysis, such as the current number of connections per node or the fastest node response time.

Load balancing calculations may be localized to each pool (member-based calculation) or they may apply to all pools of which a server is a member (node-based calculation). The variant of the mode using member-based calculation is distinguished by the extension ***least_conn_member***.

Observed

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

This is a dynamic load balancing mode, distributing connections based on various aspects of real-time server performance analysis, such as the current number of connections per node or the fastest node response time.

Load balancing calculations may be localized to each pool (member-based calculation) or they may apply to all pools of which a server is a member (node-based calculation). The variant of the mode using member-based calculation is distinguished by the extension ***observed_member***.

Predictive

Predictive mode also uses the ranking methods used by Observed mode, where nodes are rated according to a combination of the number of current connections and the response time. However, in Predictive mode, the BIG-IP analyzes the trend of the ranking over time, determining whether a node's performance is currently improving or declining. The nodes with better performance rankings that are currently improving, rather than declining, receive a higher proportion of the connections. Predictive mode works well in any environment.

This is a dynamic load balancing mode, distributing connections based on various aspects of real-time server performance analysis, such as the current number of connections per node or the fastest node response time.

Load balancing calculations may be localized to each pool (member-based calculation) or they may apply to all pools of which a server is a member (node-based calculation). The variant of the mode using member-based calculation is distinguished by the extension ***predictive_member***.

Setting the load balancing mode for a pool

A load balancing mode is specified as a pool attribute when a pool is defined and may be changed by changing this pool attribute. For information about configuring a pool, see *Working with pools*, on page 4-3. The following example describes how to configure a pool to use Ratio Member load balancing. Note that for Ratio Member mode, in addition to changing the load balancing attribute, you must assign a ratio weight to each member node.

◆ Tip

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

To configure the pool and load balancing mode using the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
   - If you are adding a new pool, click the **Add** button.
     The Add Pool screen opens.
   - If you are changing an existing pool, click the pool in the **Pools** list.
     The Pool Properties screen opens.
2. In the Add Pool screen or Pool Properties screen, configure the pool attributes. For additional information about defining a pool, click the **Help** button.

◆ Note

Round Robin is the default load balancing mode and never needs to be set unless you are returning to it from a non-default mode.

To switch a pool to *ratio_member* mode using the Configuration utility

1. In the **Current Members** list, click the member you want to edit.
2. Click the Back button (<<) to pull the member into the resources section.
3. Change or add the ratio value for the member.
4. Click the Add button (>>) to add the member back to the **Current Members** list.
5. Click **Done**.

To switch a pool to ratio_member mode from the command line

To switch a pool to **ratio_member** load balancing, use the **modify** keyword with the **bigpipe pool** command. For example, if you want to change the pool **my_pool**, to use the **ratio_member** load balancing mode and to assign each member its ratio weight, you can type the following command:

```
b pool my_pool modify { lb_method ratio_member member 11.12.1.101:80 ratio 1 member
    11.12.1.100:80 ratio 3}
```

Setting ratio weights for node addresses

The default ratio setting for any node address is **1**. If you use the Ratio (as opposed to Ratio Member) load balancing mode, you must set a ratio other than **1** for at least one node address in the configuration. If you do not change at least one ratio setting, the load balancing mode has the same effect as the Round Robin load balancing mode.

To set ratio weights using the Configuration utility

1. In the navigation pane, click **Nodes**.
2. In the Nodes list, click the Node Addresses tab.
   The Node Addresses screen opens.
3. In the Node Addresses screen, click the **Address** of the node.
   The Global Node Address screen opens.
4. In the **Ratio** box, type the ratio weight of your choice.
5. Click the **Apply** button to save your changes.

To set ratio weights from the command line

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

```
b ratio <node_ip> [<node_ip>...] <ratio weight>
```

For example, the following command sets the ratio weight to **3** for a specific node address:

```
b ratio 192.168.103.20 3
```

◆ Note

*The <**weight**> parameter must be a whole number, equal to or greater than **1**.*

Displaying ratio weights for node addresses

To display the ratio weights for all node addresses

The following command displays the current ratio weight settings for all node addresses.

```
b ratio show
```

The command displays the output shown in Figure 4.1.

```
192.168.200.51    ratio = 3
192.168.200.52    ratio = 1
```

Figure 4.1 *Ratio weights for node addresses*

To display ratio weight for specific node addresses

Use the following syntax to display the ratio setting for one or more node addresses:

```
b ratio <node_ip> [...<node_ip>] show
```

Configuring servers and the BIG-IP for Dynamic Ratio load balancing

You can configure Dynamic Ratio load balancing on RealNetworks RealServer platforms, Windows platforms equipped with Windows Management Instrumentation (WMI), or any server equipped with an SNMP agent such as the UC Davis SNMP agent or Windows 2000 Server SNMP agent.

Configuring RealNetwork RealServers

For RealNetworks, we provide a monitor plugin for the server that gathers the necessary metrics. Configuring a RealServer for Dynamic Ratio load balancing consists of four tasks:

- Installing the monitor plugin on the RealServer
- Configuring a **real_server** health check monitor on the BIG-IP
- Associating the health check monitor with the server to gather the metrics
- Creating or modifying the server pool to use Dynamic Ratio load balancing

To install the monitor plugin on the RealServer

1. Download the monitor plugin **F5RealMon.dll** from the BIG-IP. The plugin is located in **/usr/contrib/f5/isapi**. (The URL is **https://<bigip_address>/doc/rsplugin/f5realmon.dll**.)
2. Copy **f5realmon.dll** to the RealServer **Plugins** directory. (For example, **C:\Program Files\RealServer\Plugins**.)
3. If the RealServer process is running, restart it.

To configure a real_server monitor for the server node

Using the Configuration utility or the **bigpipe** command, create a health-check monitor using the **real_server** monitor template. The **real_server** monitor template is shown in the Figure 4.2.

```
monitor type real_server {
    interval 5
    timeout 16
    dest *.12345
    method "GET"
    cmd "GetServerStats"
    metrics "ServerBandwidth:1.5,CPUPercentUsage,MemoryUsage,
    TotalClientCount"
    agent "Mozilla/4.0 (compatible: MSIE 5.0; Windows NT)
    }
```

Figure 4.2 *real_server monitor template*

The **real_server** monitor template can be used as is, without modifying any of the attributes. Alternatively, you can add metrics and modify metric attribute values. To do this, you need to create a custom monitor. For example:

```
b monitor my_real_server '{ use real_server metrics "ServerBandwidth:2.0" }'
```

The complete set of metrics and metric attribute default values is shown in Table 4.3.

Metric	Default Coefficient	Default Threshold
ServerBandwidth (Kbps)	1.0	10,000
CPUPercentUsage	1.0	80
MemoryUsage (Kb)	1.0	100,000
TotalClientCount	1.0	1,000
RTSPClientCount	1.0	500
HTTPClientCount	1.0	500
PNAClientCount	1.0	500
UDPTransportCount	1.0	500
TCPTransportCount	1.0	500
MulticastTransportCount	1.0	500

Table 4.3 real_server *monitor metrics*

The metric coefficient is a factor determining how heavily the metric's value counts in the overall ratio weight calculation. The metric threshold is the highest value allowed for the metric if the metric is to have any weight at all. To understand how to use these values, it is necessary to understand how the overall ratio weight is calculated. The overall ratio weight is the sum of relative weights calculated for each metric. The relative weights, in turn, are based on three factors:

- the value for the metric returned by the monitor
- the coefficient value
- the threshold value

Given these values, the relative weight is calculated as follows:

```
w=((threshold-value)/threshold)*coefficient
```

You can see that the higher the coefficient, the greater the relative weight calculated for the metric. Similarly, the higher the threshold, the greater the relative weight calculated for any metric value that is less than the threshold. (When the value reaches the threshold, the weight goes to zero.)

Note that the default coefficient and default threshold values shown in Table 4.3 are *metric* defaults, not *template* defaults. The template defaults take precedence over the metric defaults, just as user-specified values in the custom **real_server** monitor take precedence over the template defaults. For example, in Figure 4.2, the template specifies a coefficient value of **1.5** for **ServerBandwidth** and no value for the other metrics. This means that the template will use the template default of **1.5** for the **ServerBandwidth** coefficient and the metric default of **1** for the coefficients of all other metrics. However, if a custom monitor **my_real_server** were configured specifying **2.0** as the **ServerBandwidth** coefficient, this user-specified value would override the template default.

The syntax for specifying non-default coefficient or threshold values is:

```
<metric>:<coefficient |<*>:<threshold>
```

The following examples show how to specify a coefficient value only, a threshold value only, and a coefficient and a threshold value, respectively.

```
b monitor my_real_server '{ use real_server metrics CPUPercentUsage:1.5 }'
b monitor my_real_server '{ use real_server metrics CPUPercentUsage:*:70 }'
b monitor my_real_server '{ use real_server metrics CPUPercentUsage:1.5:70 }'
```

Metric coefficient and threshold are the only non-template defaults. If a metric not in the template is to be added to the custom monitor, it must be added to the **metric** list:

```
b monitor my_real_server '{ use real_server metrics "HTTPClientCount" }'
```

To associate the monitor with the member node

Associate the custom health check monitor with the server node, creating an instance of the monitor for that node:

```
b node <node_addr> monitor use my_real_server
```

To set the load balancing method to Dynamic Ratio

Create or modify the load balancing pool to which the server belongs to use Dynamic Ratio load balancing:

```
b pool <pool_name> { lb_method dynamic_ratio  <member definition>... }
```

Configuring Windows servers with WMI

For Windows, BIG-IP provides a Data Gathering Agent **F5Isapi.dll** for the server. Configuring a Windows platform for Dynamic Ratio load balancing consists of four tasks:

- Installing the Data Gathering Agent **F5Isapi.dll** on the server
- Configuring a **wmi** health check monitor on the BIG-IP
- Associating the health check monitor with the server to gather the metrics
- Creating or modifying the server pool to use Dynamic Ratio load balancing

To install the Data Gathering Agent (F5Isapi) on the server

1. Download the **Data Gathering Agent (F5Isapi.dll)** from the BIG-IP. The plugin is located in **/usr/contrib/f5/isapi**. (The URL is **https://<bigip_address>/doc/isapi/f5isapi.dll.**)
2. Copy **f5isapi.dll** to the directory **C:\Inetpub\scripts**.
3. Open the Internet Services Manager.
4. In the left pane of the Internet Services Manager, open the folder **<machine_name>\Default Web Site\Script**, where **<machine_name>** is the name of the server you are configuring. The contents of **Scripts** folder opens in the right pane.
5. In the right pane, right click **F5Isapi.dll** and select **Properties**. The Properties dialog box for **F5Isapi.dll** opens.
6. Deselect **Logvisits**. (Logging of each visit to the agent quickly fills up the log files.)
7. Click the File Security tab.
   The File Security options appears.
8. In the **Anonymous access and authentication control group** box, click **Edit**.
   The Authentication Methods dialog box opens.
9. In the **Authentication methods** dialog box, clear all check boxes, then select **Basic Authentication**.
10. In the **Authentication methods** dialog box, click **OK** to accept the changes.
11. In the **Properties** dialog box, click **Apply**.
12. The WMI Data Gathering Agent is now ready to be used.

To configure a wmi monitor for the server node

Using the Configuration utility or the **bigpipe** command, create a health check monitor using the **wmi** monitor template. The **wmi** monitor template is shown in Figure 4.3.

```
monitor type wmi {
    interval 5
    timeout 16
    dest *:12346
    username ""
    password ""
    method "POST"
    urlpath "/scripts/F5Isapi.dll"
    cmd "GetCPUInfo, GetDiskInfo, GetOSInfo"
    metrics "LoadPercentage, DiskUsage, PhysicalMemoryUsage:1.5,
    VirtualMemoryUsage:2.0"
    post "<input type='hidden' name='RespFormat' value='HTML'>"
    agent "Mozilla/4.0 (compatible: MSIE 5.0; Windows NT)
    }
```

***Figure 4.3 wmi** monitor template*

The monitor template contains default values for all the attributes. These are template defaults. In creating a custom monitor from the template, the only default values you are required to change are the null values for username and password. For example:

```
b monitor my_wmi '{ use wmi username "dave" password "$getm" }'
```

You may also add commands and metrics and modify metric attribute values. The complete set of commands, associated metrics, and metric attribute default values are shown in Table 4.4.

Command	Metric	Default Coefficient	Default Threshold
GetCPUInfo	LoadPercentage (%)	1.0	80
GetOSInfo	PhysicalMemoryUsage (%)	1.0	80
	VirtualMemoryUsage (%)	1.0	80
	NumberRunningProcesses	1.0	100
GetDiskInfo	DiskUsage (%)	1.0	90
GetPerfCounters	TotalKBytesPerSec	1.0	10,000
	ConnectionAttemptsPerSec	1.0	500
	CurrentConnections	1.0	500
	GETRequestsPerSec	1.0	500

***Table 4.4 wmi** monitor commands and metrics*

Command	Metric	Default Coefficient	Default Threshold
	PUTRequestsPerSec	1.0	500
	POSTRequestsPerSec	1.0	500
	AnonymousUsersPerSec	1.0	500
	CurrentAnonymousUsers	1.0	500
	NonAnonymousUsersPerSec	1.0	500
	CurrentNonAnonymousUser	1.0	500
	CGIRequestsPerSec	1.0	500
	CurrentCGIRequests	1.0	500
	ISAPIRequestsPerSec	1.0	500
	CurrentISAPIRequests	1.0	500
GetWinMediaInfo	AggregateReadRate	1.0	10,000 Kbps
	AggregateSendRate	1.0	10,000 Kbps
	ActiveLiveUnicastStreams	1.0	1000
	ActiveStreams	1.0	1000
	ActiveTCPStreams	1.0	1000
	ActiveUDPStreams	1.0	1000
	AllocatedBandwidth	1.0	10,000 Kbps
	AuthenticationRequests	1.0	1000
	AuthenticationsDenied	1.0	100
	AuthorizationRequests	1.0	1000
	AuthorizationsRefused	1.0	100
	ConnectedClients	1.0	500
	ConnectionRate	1.0	500
	HTTPStreams	1.0	1000

***Table 4.4 wmi** monitor commands and metrics*

Command	Metric	Default Coefficient	Default Threshold
	HTTPStreamsReadingHeader	1.0	500
	HTTPStreamsStreamingBody	1.0	500
	LateReads	1.0	100
	PendingConnections	1.0	100
	PluginErrors	1.0	100
	PluginEvents	1.0	100
	SchedulingRate	1.0	100
	StreamErrors	1.0	100
	StreamTerminations	1.0	100
	UDPResendRequests	1.0	100
	UDPResendsSent	1.0	100

***Table 4.4 wmi** monitor commands and metrics*

For more information about the metric coefficients and thresholds, refer to the description accompanying Table 4.3, *real_server monitor metrics*, on page 4-11. Note that for a **wmi** monitor, you can add commands. To do this, simply add them to the **cmd** list.

To associate the monitor with the member node

Associate the custom health check monitor with the server node, creating an instance of the monitor for that node:

```
b node <node_addr> monitor use my_wmi
```

To set the load balancing mode to Dynamic Ratio

Use the following syntax to create or modify the load balancing pool to which the server belongs to use Dynamic Ratio load balancing:

```
b pool <pool_name> { lb_method dynamic_ratio <member definition>...}
```

Configuring SNMP servers

The BIG-IP includes an SNMP data collecting agent that can query remote SNMP agents of various types, including the UC Davis agent and the Windows 2000 Server agent. Configuring a server to use its SNMP agent for Dynamic Ratio load balancing consists of three tasks:

- Configuring a health check monitor, using either the Configuration utility or the **bigpipe** command

- Associating the health check monitor with the server to gather the metrics
- Creating or modifying the server pool to use Dynamic Ratio load balancing

BIG-IP provides two templates that you can use to create a health monitor for a server that uses an SNMP agent. These two monitor templates are:

- **snmp_dca**
  Use this template when you want to use default values or specify new values for CPU, memory, and disk metrics. When using this template, you can also specify values for other types of metrics that you wish to gather.
- **snmp_dca_base**
  Use this template when you want to use default values or specify values for metrics other than CPU, memory, and disk usage. When using this template, values for CPU, memory, and disk metrics are omitted.

◆ Note

For a description of these templates and the default values for each metric, see ***Working with templates for EAV monitors****, on page 4-142.*

Figure 4.4 shows a monitor based on the **snmp_dca** monitor template. This monitor uses the default metric values. A user can optionally specify variables for user-defined metrics.

```
monitor my_snmp_dca
    '{ use snmp_dca
    interval 10
    timeout 30
    dest *:161
    agent_type "UCD"
    cpu_coefficient "1.5"
    cpu_threshold "80"
    mem_coefficient "1.0"
    mem_threshold "70"
    disk_coefficient "2.0"
    disk_threshold "90"
    USEROID ""
    USEROID_COEFFICIENT "1.0"
    USEROID_THRESHOLD "90"
}'
```

Figure 4.4 *A monitor based on the* ***snmp_dca*** *template*

Figure 4.5 shows a monitor based on the **snmp_dca_base** monitor template. This monitor uses the default metric values.

```
monitor my_snmp_dca_base
    '{ use snmp_dca_base
    interval 10
    timeout 30
    dest *:161

    USEROID ""
    USEROID_COEFFICIENT "1.0"
    USEROID_THRESHOLD "90"
    }'
```

Figure 4.5 *A monitor based on the* ***snmp_dca_base*** *template*

◆ Note

Note that in the above examples, the user-defined variables are specified as ***USEROID****,* ***USEROID_COEFICIENT****, and* ***USEROID_THRESHOLD****. You can create any variable names you want. Although the values shown in the above examples are entered in uppercase, uppercase is not required.*

To configure a monitor based on either the **snmp_dca** or **snmp_dca_base** template, you can use either the Configuration utility or the **bigpipe** command.

◆ Note

The default agent type specified in the ***snmp_dca*** *template is UC Davis. When configuring a monitor for a Windows 2000 server, you must change the agent type to Windows 2000.*

To configure an SNMP monitor using the Configuration utility

1. In the Navigation pane, click **Monitors**.
2. Click the Add button.
   This displays the Configure Monitor Name and Parent screen.
3. Enter a unique name for the monitor in the **Name** box and select a template from the **Inherits from** box. If you want the monitor to include CPU, memory, disk, and user metrics, select the **snmp_dca** template. If you want the monitor to include user metrics only, select the **snmp_dca_base** template.
4. Click **Next**. This displays the Configure Basic Properties screen.
5. Retain or change the values in the **Interval** and **Timeout** boxes.
6. Click **Next**. This displays the Configure EAV SNMP DCA Monitor screen.

7. Retain or change the values for CPU, memory, and disk use. Also note that in the s**nmp_dca** template, the default value for the **Agent Type** property is **UCD**. To configure a monitor for a Windows 2000 agent, change this value to **WIN2000**.
8. Click **Next**.
   This displays the Configure EAV Variables screen.
9. If you are specifying user-defined metrics, configure the EAV variables by specifying a unique name and a value for each Name/Value pair.

   The three variables (that is, Name/Value pairs) correspond to OID, coefficient, and threshold. Note that if the value of the OID variable is an absolute value, verify that the user-defined threshold value is also an absolute value. If the threshold value is not absolute, BIG-IP might not factor the value into the load calculation. The default user-defined threshold value is **90**.
10. Click **Next**.
    This displays the Configure Destination Address and Service (Alias) screen. We recommend that you use the default values shown here.
11. Click **Done**.

To configure an SNMP monitor using the bigpipe command

When configuring an SNMP monitor using the **bigpipe** command, you can use the default CPU, memory, and disk coefficient and threshold values specified in the templates, or you can change the default values. Optionally, you can specify coefficient and threshold values for gathering other types of data. Note that if the monitor you are configuring is for a type of SNMP agent other than UC Davis, you must specify the agent type as an argument to the **bigpipe** command.

The following command-line examples show various ways to configure an SNMP monitor. Note that although arguments for user-defined metrics are shown in uppercase, uppercase is not required.

To configure a monitor for a UC Davis SNMP agent, using default CPU, memory, and disk use values, use the **bigpipe monitor** command, as in the following example.

```
b monitor my_snmp_dca '{ use snmp_dca }'
```

To configure a monitor for a UC Davis SNMP agent, using all default CPU, memory threshold, and disk use values and specifying a non-default memory coefficient value, use the **bigpipe monitor** command, as in the following example.

```
b monitor my_snmp_dca '{ use snmp_dca  mem_coefficient "1.5" }'
```

To configure a monitor for a UC Davis SNMP agent, using default CPU, memory threshold, and disk use values and specifying non-default memory coefficient and user values, use the **bigpipe monitor** command, as in the following example.

```
b monitor my_snmp_dca '{ use snmp_dca  mem_coefficient "1.5"/
USEROID ".1.3.6.1.4" USEROID_COEFFICIENT "1.5" USEROID_THRESHOLD/
 "80" }'
```

To configure a monitor for a UC Davis SNMP agent, omitting CPU, memory, and disk use values and using default user coefficient and user threshold values (1.0 and 90 respectively), use the **bigpipe monitor** command, as in the following example.

```
b monitor my_snmp_dca '{ use snmp_dca_base USEROID ".1.3.6.1.4"}'
```

To configure a monitor for a UC Davis SNMP agent, omitting CPU, memory, and disk use values and specifying non-default user values, use the **bigpipe monitor** command, as in the following example.

```
b monitor my_snmp_dca_base '{ use snmp_dca_base USEROID/
".1.3.6.1.4" USEROID_COEFFICIENT/ "1.5" USEROID_THRESHOLD "80" )'
```

To configure a monitor for a Windows 2000 SNMP agent, using default CPU, memory, and disk use values, use the **bigpipe monitor** command, as in the following example.

```
b monitor my_win2000_snmp_dca '{use snmp_dca agent_type "WIN2000"}'
```

To associate the health check monitor with the member node

Use the following syntax to associate the custom health check monitor with the server node and create an instance of the monitor for that node:

```
b node <node_addr> monitor use my_snmp_dca
```

To set the load balancing method to Dynamic Ratio

Use the following syntax to create or modify the load balancing pool to which the server belongs to use Dynamic Ratio load balancing:

```
b pool <pool_name> { lb_method dynamic_ratio  <member definition>... }
```

Priority-based member activation

You can load balance traffic across all members of a pool or across only members that are currently activated according to their priority number. In priority-based member activation, each member in a pool is assigned a priority number that places it in a priority group designated by that number. With all nodes available (meaning they are enabled, marked **up**, and have not exceeded their connection limit), the BIG-IP distributes connections to all nodes in the highest priority group only, that is, the group designated by the highest priority number. The **min_active_members** value determines the minimum number of members that must remain available for traffic to

be confined to that group. If the number of available nodes in the highest priority group goes below the minimum number, the BIG-IP also distributes traffic to the next higher priority group, and so on.

```
pool my_pool {
    lb_mode fastest
    min_active_members 2
    member 10.12.10.1:80 priority 3
    member 10.12.10.2:80 priority 3
    member 10.12.10.3:80 priority 3
    member 10.12.10.4:80 priority 2
    member 10.12.10.5:80 priority 2
    member 10.12.10.6:80 priority 2
    member 10.12.10.7:80 priority 1
    member 10.12.10.8:80 priority 1
    member 10.12.10.9:80 priority 1
    }
```

Figure 4.6 *Sample pool configuration for priority based member activation*

The configuration shown in Figure 4.6 has three priority groups, **3**, **2**, and **1**. Connections are first distributed to all nodes with priority **3**. If fewer than two priority **3** nodes are available, traffic is directed to the priority **2** nodes as well. If both the priority **3** group and the priority **2** group have fewer than two nodes available, traffic is directed to the priority **1** group as well. The BIG-IP continuously monitors the higher priority groups, and each time a higher priority group once again has the minimum number of available nodes, the BIG-IP again limits traffic to that group.

If you set the load balancing mode to Ratio (as opposed to Ratio Member), you must define the ratio settings for each node address.

Persistence

If you are setting up an e-commerce or other type of transaction-oriented site, you may need to configure persistence on the BIG-IP. Persistence is one of the pool attributes listed in Table 4.1.

Whether you need to configure persistence or not simply depends on how you store client-specific information, such as items in a shopping cart, or airline ticket reservations. For example, you may store the airline ticket reservation information in a back-end database that all nodes can access, or on the specific node to which the client originally connected, or in a cookie on the client’s machine.

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

The BIG-IP tracks information about individual persistent connections, and keeps the information only for a given period of time. The way in which persistent connections are identified depends on the type of persistence.

Types of persistence

The types of persistence are:

- **Simple persistence**
  Simple persistence supports TCP and UDP protocols, and tracks connections based only on the client IP address.
- **HTTP cookie persistence**
  HTTP cookie persistence uses an HTTP cookie stored on a client's computer to allow the client to reconnect to the same server previously visited at a web site.
- **SSL persistence**
  SSL persistence is a type of persistence that tracks SSL connections using the SSL session ID. Even when the client's IP address changes, the BIG-IP still recognizes the connection as being persistent based on the session ID.
- **SIP Call-ID persistence**
  SIP persistence is a type of persistence used for proxy servers that receive Session Initiation Protocol (SIP) messages sent through UDP. SIP is a protocol that enables real-time messaging, voice, data, and video.
- **Destination address affinity (sticky persistence)**
  Destination address affinity directs requests for a certain destination to the same proxy server, regardless of which client the request comes from.
- **WTS persistence**
  Windows Terminal Server (WTS) persistence tracks and load balances connections between WTS client-server configurations.

◆ Note

All persistence methods are properties of pools.

Persistence options

When setting up persistence, you can enable either of the following two options:

- **Persistence across virtual servers with the same address**
  Persistence across virtual servers with the same address causes the BIG-IP to maintain persistence only when the virtual server hosting the connection has the same virtual address as the virtual server hosting the initial persistent connection.
- **Persistence across all virtual servers**
  Persistence across all virtual servers causes the BIG-IP to maintain persistence for all connections requested by the same client, regardless of which virtual server hosts each individual connection initiated by the client.

Simple persistence

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

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

Persistence settings for pools apply to all protocols. When the persistence timer is set to a value greater than 0, persistence is **on**. When the persistence timer is set to 0, persistence is **off**.

To configure simple persistence for pools using the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. Select the pool for which you want to configure simple persistence.
   The Pool Properties screen opens.
3. Click the Persistence tab.
   The Persistence Properties screen opens.
4. In the Persistence Type section, click the **Simple** button.
   Type the following information:
   - **Timeout (seconds)**
     Set the number of seconds for persistence on the pool. (This option is not available if you are using rules.)
   - **Mask**
     Set the persistence mask for the pool. The persistence mask determines persistence based on the portion of the client's IP address that is specified in the mask.
5. Click the **Apply** button.

To configure simple persistence for pools from the command line

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

```
b pool <pool_name> modify { \
persist_mode simple \
simple_timeout <timeout> \
simple_mask <ip_mask> }
```

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

```
b pool my_pool modify { \
persist_mode simple \
simple_timeout 3600 \
simple_mask 255.255.255.0 }
```

Using a simple timeout and a persist mask on a pool

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

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

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

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

The complete syntax for the command is:

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

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

```
b pool my_pool modify { \
persist_mode simple \
simple_timeout 1200 \
simple_mask 255.255.255.0 }
```

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

```
b pool my_pool modify { simple_mask none }
```

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

```
b pool my_pool persist show
```

HTTP cookie persistence

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

There are four types of cookie persistence available:

- Insert mode
- Rewrite mode
- Passive mode
- Hash mode

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

Insert mode

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

To activate Insert mode using the Configuration utility

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

To activate Insert mode from the command line

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

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

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

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

Rewrite mode

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

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

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

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

◆ Note

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

To activate Rewrite mode cookie persistence using the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the **Pools** list, click the pool for which you want to set up Rewrite mode.
   The properties screen for the pool you clicked opens.
3. Click the Persistence tab.
   The Persistence screen opens.
4. Click the **Active HTTP Cookie** button.
5. Select **Rewrite** mode from the **Method** list.

6. Type the timeout value in days, hours, minutes, and seconds. This value determines how long the cookie lives on the client computer before it expires.
7. Click the **Apply** button.

To activate Rewrite mode cookie persistence from the command line

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

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

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

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

Passive mode

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

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

```
Header add Set-Cookie: "BIGipServer my_pool=184658624.20480.000; expires=Sat, 19-Aug-2002
    19:35:45 GMT; path=/"
```

In this example, **my_pool** is the name of the pool that contains the server node, **184658624** is the encoded node address and **20480** is the encoded port. You can generate a cookie string with encoding automatically added using the **bigpipe makecookie** command:

```
b makecookie <server_address:service> [ > <file> ]
```

The command above prints a cookie template, similar to the following two examples below, to the screen or the redirect file specified.

```
Set-Cookie:BIGipServer[poolname]=336268299.20480.0000; path=/

Set-Cookie:BIGipServer[poolname]=336268299.20480.0000; expires=Sat, 01-Jan-2002 00:00:00
    GMT; path=/
```

To create your cookie from this string, type the actual pool names and the desired expiration date and time.

Alternatively, you can perform the encoding using the following equation for address (a.b.c.d):

```
d*(256^3) + c*(256^2) + b*256 +a
```

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

To activate Passive mode cookie persistence using the Configuration utility

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

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

To activate Passive mode cookie persistence from the command line

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

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

◆ Note

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

Hash mode

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

To configure the cookie persistence hash option using the Configuration utility

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

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the Pools list, click the pool for which you want to set up hash mode persistence.
   The properties screen for the pool you clicked opens.
3. Click the Persistence tab.
   The Persistence screen opens.
4. Click the **Cookie Hash** button.
   Set the following values (see the following Table 4.5 for more information):
   - **Cookie Name**
     Type the name of an HTTP cookie being set by the Web site. This could be something like Apache or SSLSESSIONID. The name depends on the type of web server your site is running.
   - **Hash Values**
     The **Offset** is the number of bytes in the cookie to skip before calculating the hash value. The **Length** is the number of bytes to use when calculating the hash value.
5. Click the **Apply** button.

To configure the hash cookie persistence option from the command line

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

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

The **<cookie_name>**, **<cookie_value_offset>**, and **<cookie_value_length>** values are described in Table 4.5.

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

Table 4.5 *The cookie hash mode values*

SSL persistence

SSL persistence is a type of persistence that tracks SSL connections using the SSL session ID, and it is a property of each individual pool. Using SSL persistence can be particularly important if your clients typically have translated IP addresses or dynamic IP addresses, such as those that Internet service providers typically assign. Even when the client's IP address changes, the BIG-IP still recognizes the connection as being persistent based on the session ID.

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

You can set up SSL persistence from the command line or using the Configuration utility. To set up SSL persistence, you need to do two things:

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

◆ Note

Do not enable SSL persistence on pools that load balance plain-text traffic, that is, traffic resulting from SSL proxies on which SSL termination is enabled.

To activate SSL persistence from the command line

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. Click the appropriate pool in the list.
   The Pool Properties screen opens.
3. Click the Persistence tab.
   The Persistence screen opens.

4. Click the **SSL** button.
5. In the **Timeout** box, type the number of seconds that the BIG-IP should store SSL session IDs before removing them from the system.
6. Click the **Apply** button.

To activate SSL persistence from the command line

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

```
b pool <pool_name> modify { persist_mode ssl ssl_timeout <timeout> }
```

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

```
b pool my_pool modify { persist_mode ssl ssl_timeout 3600 }
```

To display persistence information for a pool

To show the persistence configuration for the pool:

```
b pool <pool_name> persist show
```

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

```
b pool my_pool persist show
```

SIP Call-ID persistence

Session Initiation Protocol (SIP) is an application-layer protocol that manages sessions consisting of multiple participants, thus enabling real-time messaging, voice, data, and video. With SIP, applications can communicate with one another by exchanging messages through TCP or UDP. Examples of such applications are internet conferencing and telephony, or multimedia distribution.

SIP Call-ID persistence is a new type of persistence available for server pools. You can configure Call-ID persistence for proxy servers that receive Session Initiation Protocol (SIP) messages sent through UDP.

◆ Note

BIG-IP currently supports persistence for SIP messages sent through UDP only.

When activating SIP Call-ID persistence for a server pool, you can specify the following:

- The name of the server pool (required)
- A timeout value for persistence records (optional)

  This timeout value allows the BIG-IP to free up resources associated with old SIP persistence entries, without having to test each inbound

packet for one of the different types of SIP final messages. A default timeout value exists, which is usually 32 seconds. This timeout value is the window of time that a stateful proxy maintains state. If you change the timeout value, we recommend that the value be no lower than the default.

To activate SIP Call-ID persistence, you can use either the Configuration utility or the **bigpipe pool** command.

To activate SIP persistence using the Configuration utility

1. Start the Configuration utility.
2. In the Navigation pane, click **Pools**.
   The Pools screen opens.
3. Select a pool name.
4. Click the Persistence tab.
5. Click the button for SIP persistence.
6. Click the **Apply** button.

To activate SIP persistence from the command line

Use the following syntax to activate SIP Call-ID persistence from the command line.

```
bigpipe pool <pool name> { persist_mode sip [sip_timeout <timeout>] }
```

To display the contents of the hash table

To display the contents of the SIP persistence hash table, use the **bigpipe** command as follows:

```
bigpipe sip dump
```

Destination address affinity (sticky persistence)

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

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

To activate destination address affinity using the Configuration utility

You can only activate destination address affinity on pools directly or indirectly referenced by wildcard virtual servers. For information on setting up a wildcard virtual server, see the *Wildcard virtual servers*, on page 4-71. Follow these steps to configure destination address affinity:

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

To activate destination address affinity from the command line

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

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

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

```
b pool <pool_name> sticky clear
```

To show the persistence configuration for the pool:

```
b pool <pool_name> persist show
```

WTS persistence

Windows Terminal Server (WTS) is a Windows feature that allows a Windows 2000 client to run Windows remotely from a Windows.NET.Server system over a TCP connection. To track and load balance connections between WTS dient and server systems, BIG-IP offers a type of persistence known as ***WTS persistence***.

To activate WTS persistence, you must enable it as a BIG-IP pool attribute. You must also configure a WTS server, in either of two modes: hash mode or passive mode.

◆ Note

For information on how to configure a WTS server for BIG-IP WTS persistence, see your Windows.NET.Server product documentation.

Hash mode

When configured in hash mode, a WTS server does not participate in a session directory; that is, the server cannot share sessions with other WTS servers. ***Hash mode*** ensures that WTS clients provide data to the BIG-IP to allow the BIG-IP to consistently connect that client to the same WTS server. If the client data that BIG-IP requires is not provided, BIG-IP load balances the connection according to the way that the user has configured BIG-IP for load balancing.

Passive mode

When configured in passive mode, a WTS server participates in a session directory, that is, the server can share sessions with other WTS servers. ***Passive mode*** provides a way for BIG-IP to maintain persistent connections when WTS servers are clustered together. Normally, WTS servers, when participating in a session directory, map WTS clients to their appropriate servers. If a client connects to the wrong server in the cluster, the targeted server checks its client-server mapping and performs a rewrite to the correct server.

When configured in passive mode, however, a WTS server always rewrites the connection to the same BIG-IP virtual server, instead of to the correct WTS server directly. The BIG-IP then sends the connection to the correct WTS server.

Activating WTS persistence on BIG-IP

To activate WTS persistence on BIG-IP, you must perform three tasks:

- Enable TCP service 3389
- Create a pool containing all cluster members and specifying the **msrdp** persistence type
- Create a virtual server that uses the pool

To enable TCP service 3389 from the command line

To enable TCP service 3389, use the following command:

```
b service 3389 tcp enable
```

To create a pool with the WTS persistence attribute from the command line

To create a pool that is configured for WTS persistence and that contains two members of a WTS cluster, use the **bigpipe pool** command as in the following example:

```
b pool my_cluster_pool persist_mode msrdp { member 11.12.1.101:3389 member
    11.12.1.100:3389 }
```

To create a virtual server from the command line

To create a virtual server that uses the pool **my_cluster_pool**, use the **bigpipe virtual** command as in the following example:

```
b virtual 192.200.100.25:3389 use pool my_cluster_pool
```

Maintaining persistence across virtual servers that use the same virtual addresses

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

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

```
b virtual v1:http use pool http_pool
b virtual v1:ssl use pool ssl_pool
b virtual v2:ssl use pool ssl_pool
```

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

For example, a client makes an initial connection to **v1:http** and the load balancing mechanism assigned to the pool **http_pool** chooses **n1:http** as the node. If the same client then connects to **v2:ssl**, the BIG-IP starts tracking a new persistence session, and it uses the load balancing mode to determine which node should receive the connection request because the requested virtual server uses a different virtual address (**v2**) than the virtual server hosting the first persistent connection request (**v1**). In order for this mode to be effective, virtual servers that use the same virtual address, as well as those that use TCP or SSL persistence, should include the same node addresses in the virtual server mappings.

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

1. In the navigation pane, click **System**.
   The Network Map screen opens.
2. Click the Advanced Properties tab.
   The BIG-IP System Control Variables screen opens.

3. Click the **Allow Persistence Across All Ports for Each Virtual Address** check box. (To disable this persistence mode, clear the check box).
4. Click the **Apply** button.

To activate persistence for virtual servers that use the same address from the command line

The global variable **persist_across_services** turns this mode on and off. To activate the persistence mode, type:

```
b global persist_across_services enable
```

To deactivate the persistence mode, type:

```
b global persist_across_services disable
```

Maintaining persistence across all virtual servers

You can set the BIG-IP to maintain persistence for all connections requested by the same client, regardless of which virtual server hosts each individual connection initiated by the client. When this mode is turned on, the BIG-IP attempts to send all persistent connection requests received from the same client, within the persistence time limit, to the same node. Connection requests from the client that do not use persistence are load balanced according to the currently selected load balancing mode.

The following examples show virtual server mappings, where the virtual servers **v1:http** and **v2:http** reference the **http1_pool** and **http2_pool** (both pools contain the nodes **n1:http** and **n2:http**) and the virtual servers **v1:ssl** and **v2:ssl** reference the pools **ssl1_pool** and **ssl2_pool** (both pools contain the nodes **n1:ssl** and **n2:ssl**). Each virtual server uses persistence:

```
b virtual v1:http use pool http1_pool
b virtual v1:ssl use pool ssl1_pool
b virtual v2:http use pool http2_pool
b virtual v2:ssl use pool ssl2_pool
```

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

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

To activate persistence across all virtual servers using the Configuration utility

1. In the navigation pane, click the **System** icon.
   The Network Map screen opens.
2. Click the Advanced Properties tab.
   The BIG-IP System Control Variables screen opens.
3. Click the **Allow Persistence Across All Virtual Servers** check box to activate this persistence mode.
4. Click the **Apply** button.

To activate persistence across all virtual servers from the command line

The global variable **persist_across_virtuals** turns this mode on and off. To activate the persistence mode, type:

```
b global persist_across_virtuals enable
```

To deactivate the persistence mode, type:

```
b global persist_across_virtuals disable
```

HTTP redirection

Another attribute of a pool is HTTP redirection. ***HTTP redirection*** allows you to configure a pool so that HTTP traffic is redirected to another protocol identifier, host name, port number, or URI path. For example, if all members of a pool are unavailable (that is, the members are disabled, marked **down**, and have exceeded their connection limit), the HTTP request can be redirected to the fallback host, with the HTTP reply Status Code **302 Found**.

When configuring a pool to redirect HTTP traffic to a fallback host, you can use an IP address or a fully-qualified domain name (FQDN), or you can use a special format string included in the BIG-IP. These format strings can also be used for specifying protocol identifiers, ports, and URIs.

The following two sections describe these two ways of redirecting HTTP requests. Following these two sections is a description of a related feature, which allows you to configure a server to rewrite the specified HTTP redirection.

Using IP addresses and Fully Qualified Domain Names

When redirecting traffic to a fallback host, you can specify the fallback host as an IP address or as a fully qualified domain name (FQDN). In either case, it may include a port number. The example in Figure 4.7 redirects the request to **http://redirector.sam.com**.

```
pool my_pool {
    member 10.12.10.1:80
    member 10.12.10.2:80
    member 10.12.10.3:80
    fallback redirector.sam.com
}
```

Figure 4.7 *Fallback host in a pool*

◆ Note

The HTTP redirect mechanism is not a load balancing method. The redirect URL may be a virtual server pointing to the requested HTTP content, but this is not implicit in its use.

Table 4.6 shows how different fallback host specifications are resolved.

Requested URL	Fallback Host Specification	Redirect URL
http://www.sam.com/	fallback.sam.com	http://falback.sam.com/
http://www.sam.com/	fallback.sam.com:8002	http://fallback.sam.com:8002/
http://www.sam.com:8001	fallback.sam.com	http://fallback.sam.com/
http://www.sam.com:8001/	fallback.sam.com:8002	http://fallback.sam.com:8002/
http://www.sam.com/sales	fallback.sam.com	http://fallback.sam.com/sales
http://192.168.101.3/	fallback.sam.com	http://fallback.sam.com/
http://192.168.101.3/sales	fallback.sam.com	http://fallback.sam.com/sales
http://www.sam.com/sale	192.168.101.5	http://192.168.101.5/sales
http://192.168.101.3/sales/default.asp?q=6	fallback.sam.com	http://fallback.sam.com/sales/default.asp?q=6

Table 4.6 *How the fallback host specifications are resolved*

Using format strings (expansion characters)

To allow HTTP redirection to be fully configurable with respect to target URI, the following format strings are available. These strings can be used within both pools and rules. (For more information on using HTTP redirection format strings within rules, see *Pool selection based on HTTP header data*, on page 4-56.)

Table 4.7 lists and defines the format strings that you can use to specify HTTP redirection.

Format String	Description
%h	host name, as obtained from the Host: header of the client
%p	Port, from the virtual server listening port
%u	URI path, as obtained from a GET/POST request

Table 4.7 *Format strings for HTTP redirection*

An example of a fallback host string is **https://%h/sample.html**. In this string, specifying **https** as the protocol identifier causes the traffic to be redirected to that protocol instead of the standard **http** protocol. Also, the string **sample.html** causes the traffic to be redirected to that URI instead of to the standard URI specified in the HTTP header, which would normally be represented in the fallback string as **%u**.

Table 4.8 shows some sample redirection specifications, their explanations, and their resulting redirection.

Redirection string	Explanation	Resulting redirection
%h:%p/%u	No redirection (preserve host name, port, and path)	http://www/example.com:8080/sample
%h/unavailable	change path, remove port	http://www/example.com/unavailable
https://%h/unavailable	Specify **https** as protocol, remove port, change path	https://www/example.com/unavailable
www.sample.com:8080/%u	Change host name and port, preserve path	http://www.sample.com:8080/sample
https://1.2.3.4:443/%u/unavilable.html	Specify **https** as protocol, change host name, port, and path	https://1.2.3.4:443/sample/unavailable.html
ftp://1.2.3.4:%p/unavailable/%u	Specify **ftp** as protocol, change host name and path	ftp://1.2.3.4:8080/unavailable/sample
rtsp://%h:554/streamingmedia/%u	Specify **rtsp** as protocol, change port and path	rtsp://www.example.com:554/streamingmedia /sample

Table 4.8 *Sample HTTP redirections using format strings*

The example in Figure 4.8 shows a pool configured to redirect an HTTP request to a different protocol (**https**) host name (**1.2.3.4**), port number (**443**), and path (**unavailable.html**).

```
pool my_pool {
member 10.12.10.1:80
member 10.12.10.2:80
member 10.12.10.3:80

https://1.2.3.4:443/%u/unavailable.html
}
```

Figure 4.8 *HTTP redirection specified in a pool*

Rewriting HTTP redirection

Sometimes, a client request is redirected from the HTTPS protocol to the HTTP protocol, which is a non-secure channel. If you want to ensure that the request remains on a secure channel, you can cause that redirection to be rewritten so that it is redirected back to the HTTPS protocol. Also, through the rewriting of redirections, you can rewrite a port number or a URI path.

You can rewrite HTTP redirections in either of two ways:

- You can create an SSL Accelerator proxy and configure the rewriting of HTTP redirections as a proxy option. For more information, see *Rewriting HTTP redirection*, on page 4-105.
- If your web server is an IIS server, you can configure that server, instead of your SSL proxy, to rewrite any HTTP redirections. Part of this IIS server configuration includes the installation of a special BIG-IP filter, **redirectfilter.dll**, on the IIS server. The following section provides this IIS configuration procedure.

To install the filter for rewriting HTTP redirection

To install the ISAPI filter (**redirectfilter.dll**) for use with a Microsoft Internet Information Server (IIS) version 4.0 or 5.0, follow these steps:

1. Copy the filter DLL to an appropriate folder, such as the SCRIPTS or CGI-BIN subdirectory.
2. Open the Internet Service Manager (MMC).
3. Select the appropriate level for the ISAPI filter:
   a) If you intend to use the ISAPI filter with all Web sites, select the **ServerName** icon.
   b) If you intend to use the ISAPI filter with a specific Web site, select the icon for that Web site (for example, the default Web site).
4. Right-click the level (icon) that you selected.
5. Click the **ISAPI Filters** tab.

   Note: *To configure an ISAPI filter for all Web sites, first click the* ***Edit*** *button that is next to the Master Properties of the WWW Service.*
6. Click **Add**.
7. Type a name for the ISAPI filter.
8. Click **Browse** and select the ISAPI filter that you copied in step 1.
9. Click **OK**.
10. Stop the IISADMIN service. To do this, either type **net stop iisadmin /y** at a command prompt, or use the Services applet that is located in Control Panel (in Windows NT 4.0) or Administrative Tools (in Windows 2000).

11. Start the World Wide Web Publishing Service by typing **net start w3svc** at a command prompt, or by using the Services applet that is located in Control Panel (in Windows NT 4.0) or Administrative Tools (in Windows 2000).
12. Repeat the previous step for any other services that were stopped in step 11.
13. Browse back to the **ISAPI Filters** tab (by following steps 1-5) and verify that the filter is loaded properly. You should see a green arrow that is pointing up under the Status column.

◆ Note

*The **ISAPI Filters** tab specifies a load order, with the filter at the top of the list loading first. Normally **Sspifilt.dll**, the ISAPI filter for SSL, is at the top of the list to allow any other filters to access data before IIS encrypts and transmits or decrypts and receives TTPS traffic.*

HTTP header insertion

An optional attribute of a pool is HTTP header insertion. Using this attribute, you can configure a pool to insert a header into an HTTP request. The HTTP header being inserted can include a client IP address. Including a client IP address in an HTTP header is useful when a connection goes through a secure network address translation (SNAT) and you need to preserve the original client IP address.

The header insertion must be specified in the pool definition as a quoted string. Figure 4.9 shows the required syntax.

```
pool <pool_name> {
  header insert <quoted string>
}
```

***Figure 4.9** Syntax of a header insertion string within a pool*

Optionally, you can include rule variables in the quoted string. For example, the pool definition shown in 4.10 uses the rule variable **client_addr** to represent the original client IP address of an HTTP request.

```
pool my_pool {
  header insert "OrigClientAddr:${client_addr}"
  member 10.0.0.1:80
  member 10.0.0.2:80
  member 10.0.0.3:80
}
```

***Figure 4.10** Example of a rule variable within a pool for header insertion*

The rule variables that can be used for header insertion are:

- **client_addr**
- **client_port**
- **server_addr**
- **server_port**
- **link_qos**
- **ip_qos**

Figure 4.11 shows a pool that inserts a header, using all of the above rule variables.

```
pool my_pool {
  header insert "ClientSide:${client_addr}:${client_port} ->
${server_addr}:${server_port} tos=${ip_tos} qos=${link_qos}"
  member 10.0.0.1:80
  member 10.0.0.2:80
  member 10.0.0.3:80
}
```

Figure 4.11 *Pool with header insertion string using multiple rule variables*

The above header insertion string inserts a header such as that shown in Figure 4.12 into an HTTP request:

```
GET /index.html HTTP/1.0
ClientSide: 10.0.0.1:3340 -> 10.0.0.101:80 tos=16 qos=0
Host: www.yahoo.com
Connection: Keep-Alive
```

Figure 4.12 *Header resulting from a header insertion string within a pool*

◆ Note

If the rule variable specified is not a valid variable, the invalid variable name is inserted directly into the HTTP request, with no substitution.

In addition to inserting a client IP address into an HTTP request, you can configure an SSL Accelerator proxy to insert other types of headers into HTTP requests. Examples of headers that an SSL proxy can insert are: information on client certificates, cipher specifications, and client session IDs.

For more information on rule variables and on configuring an SSL proxy to insert headers into HTTP requests, see *Rule-based pool selection*, on page 4-50 and *Inserting headers into HTTP requests*, on page 4-95.

Quality of Service (QoS) level

Another attribute of a pool is the Quality of Service (QoS) level. The ***QoS*** level is a means by which network equipment can identify and treat traffic differently based on an identifier. As traffic enters the site, the BIG-IP can set the QoS level on a packet, based on the QoS level defined in the pool to which the packet is sent. The BIG-IP can also apply a rule that sends the traffic to different pools of servers based on the Quality of Service level.

The BIG-IP can tag outbound traffic (the return packets based on an **HTTP GET**) based on the QoS value set in the pool. That value is then inspected by upstream devices and given appropriate priority. Based on a rule, the BIG-IP can examine incoming traffic to see if it has a particular QoS or ToS tag in the header. The BIG-IP can then make a rule-based load balancing decision based on that tag.

Figure 4.13 shows how to configure a pool so that a QoS level is set for a packet sent to that pool. In this example, the QoS tag, represented by the ***link_qos*** variable, is set to **3** when sending packets to the client, and set to **4** when packets are sent to the server.

```
pool http_pool {
  link_qos to client 3
  link_qos to server 4
}
```

Figure 4.13 *Example of a pool that sets the QoS level on a packet*

In addition to configuring a pool to set the QoS level on a packet, you can configure a rule that selects a pool based on the existing QoS value within the packet. For more information, see *Quality of Service (QoS) level*, on page 4-53.

Type of Service (ToS) level

Another attribute of a pool is the Type of Service (ToS) level. The ***ToS*** level, also known as the ***DiffServ*** value, is a means by which network equipment can identify and treat traffic differently based on an identifier. As traffic enters the site, the BIG-IP can set the ToS level on a packet, based on the ToS level defined in the pool to which the packet is sent. The BIG-IP can also apply a rule and send the traffic to different pools of servers based on the ToS level.

The BIG-IP can tag outbound traffic (the return packets based on an **HTTP GET**) based on the ToS value set in the pool. That value is then inspected by upstream devices and given appropriate priority. Based on a rule, the BIG-IP can examine incoming traffic to see if it has a particular ToS tag in the header. The BIG-IP can then make a rule-based load balancing decision based on that tag.

Figure 4.14 shows how to configure a pool so that a ToS level is set for a packet sent to that pool. In this example, the ToS tag, represented by the ***ip_tos*** variable, is set to **16** when sending packets to the client, and set to **16** when packets are sent to the server.

```
pool http_pool {
  ip_tos to client 16
  ip_tos to server 16
}
```

Figure 4.14 *Example of a pool that sets the ToS level on a packet*

In addition to configuring a pool that sets the ToS level on a packet, you can configure a rule that selects a pool based on the existing ToS value within the packet. For more information, see *IP Type-Of-Service (ToS) level*, on page 4-53.

Disabling SNAT and NAT connections

When configuring a pool, you can specifically disable any secure network address translations (SNATs) or network address translations (NATs) for any connections that use that pool. By default, this setting is enabled.

For general information on SNATs and NATs, see *Address translation: SNATs, NATs, and IP forwarding*, on page 4-122.

The example in Figure 4.15 shows the syntax for disabling SNAT and NAT translation for any connections that use the pool **my_pool**.

```
pool my_pool {
   snat disable
   nat disable
   member 10.1.1.1:80
   member 10.1.1.2:80
   }
```

Figure 4.15 *Disabling SNAT and NAT translations*

To disable a SNAT or NAT connection for a pool using the Configuration utility

1. In the navigation pane, click **Pools**.
2. Click the **Add** button.
3. Clear the **Enable SNATs** check box. (By default, this box is checked.)
4. Click **Done**.

To disable a SNAT or NAT connection for a pool from the command line

```
b pool <pool_name> modify { snat disable }
```

One case in which you might want to configure a pool to disable SNAT or NAT connections is when you want the pool to disable SNAT or NAT connections for a specific service. In this case, you could create a separate pool to handle all connections for that service, and then configure the **snat disable** or **nat disable** attribute on that pool. The following section describes this procedure.

To disable SNAT connections that use a specific service

The following procedure creates an automapped SNAT for a VLAN, creates a pool that disables SNAT or NAT connections, and then directs a wildcard virtual server using port **162** to send connections to the newly-defined pool.

1. Enable SNAT automapping on the self IP address for VLAN **my_vlan**. For example:

   ```
   b self 192.168.33.14 vlan my_vlan snat automap enable
   ```

2. Create an automapped SNAT for the VLAN **my_vlan**. For example:

   ```
   b snat map my_vlan to auto
   ```

3. Create a forwarding pool with the **snat disable** attribute defined. For example:

   ```
   b pool snat_disable_pool { snat disable forward }
   ```

   Note: For information on forwarding pools, see **Forwarding pools**, on page 4-47.

4. Create a wildcard virtual server for the VLAN **my_vlan**, specifying port **162** and the pool **snat_disable_pool**. For example:

   ```
   b virtual my_vlan :162 use pool snat_disable_pool
   ```

Figure 4.16 shows the resulting entries in the /**config/bigip.conf** file.

```
# self IP addresses
self 192.168.33.14 {
vlan my_vlan
netmask 255.255.255.0
broadcast 192.168.33.255
snat automap enable
}

# server pools
pool snat_disable_pool {
snat disable
forward
}

# virtual servers
virtual servers:162 unit 1 {
use pool snat_disable_pool
translate addr disable
```

Figure 4.16 *Sample entries in the* ***/config/bigip.conf*** *file*

Figure 4.17 shows an example of a rule that sends SNAT connections to a pool that disables SNAT connections on a range of ports, defined in the class **IP_Port_Range**.

```
# The snat_disable pool disables all SNAT connections.
if (client_port == one of IP_Port_Range {
   use ( snat_disable)
}
else {
   use ( other_pool)
}

# The IP_Port_Range class contains a list of two
ports/services.

class IP_Port_Range {
    161
    162
}
```

Figure 4.17 *A rule that disables SNAT connections for a range of ports*

Forwarding pools

A ***forwarding pool*** is a pool that specifies that a connection should be forwarded, using IP routing, instead of load balanced. In many cases, this eliminates the need to create a forwarding virtual server.

Forwarding pools are typically used with wildcard virtual servers or network virtual servers only. When you enable forwarding on a pool, you can apply any feature that can be configured on a pool to a forwarding connection.

A pool configured for forwarding has no members. Also, this type of pool cannot be the default gateway pool.

Figure 4.18 shows an example of a pool configured for forwarding.

```
pool my_pool {
   link_qos to client 5
   link_qos to server 5
   forward
}
```

Figure 4.18 *Example of a pool configured for forwarding*

To configure a pool for forwarding using the Configuration utility

1. In the navigation pane, click **Pools**.
2. Click the **Add** button.
3. Click the **forwarding** button. If you enable forwarding, you cannot enter a list of pool members.
4. Click **Done**.

To configure a pool for forwarding from the command line

```
b pool <pool_name> { forward }
```

◆ Note

If you want to enable IP forwarding for a virtual server or globally for the BIG-IP, see Forwarding virtual servers, on page 4-76 and IP forwarding, on page 4-135, respectively.

Rules

As described in the *Pools* section, a pool may be referenced directly by the virtual server, or indirectly through a ***rule***, which chooses among two or more load balancing pools. In other words, a rule selects a pool for a virtual server. A rule is referenced by a 1- to 31-character name. When a packet arrives that is destined for a virtual server that does not match a current connection, the BIG-IP can select a pool by evaluating a virtual server rule. The rule is configured to ask true or false questions such as:

- HTTP header load-balancing: Does the packet data contain an HTTP request with a URI ending in **cgi**?
- IP header load balancing: Does the source address of the packet begin with the octet **206**?

In addition to creating a rule to select a pool, you can also create a rule to redirect an HTTP request to a specific host name, port number, or URI path.

The remainder of this section covers the following topics:

- Rule-based pool selection - Describes how to use the pool-selection criteria to configure load balancing.
- Rule-based HTTP redirection - Describes how to create a rule to redirect an HTTP request to a specific host name, port, or URI.
- Rule Statements - Describes the various types of expressions and operands that you are allowed to use when constructing rules.
- Configuring rules - Provides the procedure for configuring a rule, using either the Configuration utility or the command-line interface.
- Additional rule examples - Provides additional examples showing how to configure rules for various types of pool selection.

◆ Note

Once you have created a rule, you need to configure a virtual server to reference that rule. For information on configuring a virtual server to reference a rule, see ***Configuring virtual servers that reference rules****, on page 4-82.*

Rule-based pool selection

Table 4.9 lists the various criteria you can use when creating a rule to select a pool.

Pool-selection criteria	Description
Pool selection based on HTTP request data	You can send connections to a pool or pools based on HTTP header information you specify.
Pool selection based on IP packet header data	You can send connections to a pool or pools based on IP addresses, port numbers, IP protocol numbers, Quality of Service (Qos), and Type of Service (ToS) levels defined within a packet.
Pool selection based on ***one of*** operator	You can send connections to a pool or pools based on whether the destination address is a member of a specific named class, such as ***one of*** AOL.
Pool selection based on HTTP header data (Cache rule)	This type of rule is any rule that contains a cache statement. A cache rule selects a pool based on HTTP header data. You cannot use it with FTP.

Table 4.9 *The attributes you can configure for a rule*

The following sections describe specific rule statements that you can use to select pools for load balancing.

◆ Note

You must define a pool before you can define a rule that references the pool.

Pool selection based on HTTP request data

A rule specifies what action the BIG-IP takes depending on whether a question is answered **true** or **false**. A rule may either select a pool or ask another question. For example, you may want a rule that logically states: "If the packet data contains an HTTP request with a URI ending in **cgi**, then load balance using the pool **cgi_pool**. Otherwise, load balance using the pool **default_pool**".

Figure 4.19 shows a rule with an HTTP request variable that illustrates this example.

```
rule cgi_rule {
    if (http_uri ends_with "cgi") {
       use ( cgi_pool )
    }
    else {
       use ( default_pool )
    }
}
```

Figure 4.19 *A rule based on an HTTP header variable*

Rules normally run right after the BIG-IP receives a packet that does not match a current connection. However, in the case of an HTTP request, the first packet is a TCP SYN packet that does not contain the HTTP request. In this case, the BIG-IP proxies the TCP handshake with the client and begins evaluating the rule again when the packet containing the HTTP request is received. When a pool has been selected and a server node selected, the BIG-IP proxies the TCP handshake with the server node and then passes traffic normally.

For examples of rules that select pools based on header information inserted into HTTP requests by an SSL Accelerator proxy, see *Inserting headers into HTTP requests*, on page 4-95.

Pool selection based on IP packet header data

In addition to specifying the HTTP variables within a rule, you can also select a pool by specifying IP packet header information within a rule. The types of information you can specify in a rule are:

- Client IP address
- Server IP address
- Client port number
- Server port number
- IP protocol number
- QoS level
- ToS level

To specify IP packet header variables within a rule using the Configuration utility

1. In the navigation pane, click **Rules**.
2. Click the **Add** button.
3. In the **Name** box, enter a unique name for the rule.
4. In the **Type** box, click the button for **Rule Builder**.
5. Click **Next**.
6. Select a variable on the left side of the screen.
7. Fill in all information within the row.
8. Click **Next**.
9. Select **Discard**.
10. Click **Next**.
11. Select **No Action** or **Discard** from the box.
12. Click **Done**.

The following sections describe the specific types of IP packet header data that you can specify within a rule.

IP addresses

You can specify the **client_addr** or the **server_addr** variable within a rule to select a pool. For example, if you want to load balance based on part of the client's IP address, you might want a rule that states:

"All client requests with the first byte of their source address equal to **206** will load balance using a pool named **clients_from_206** pool. All other requests will load balance using a pool named **other_clients_pool**."

Figure 4.20 shows a rule that implements the preceding statement.

```
rule clients_from_206_rule {
    if ( client_addr equals 206.0.0.0 netmask 255.0.0.0 ) { {
        use ( clients_from_206 )
     }
      else {
         use ( other_clients_pool )
       }
}
```

Figure 4.20 *A rule based on the client IP address variable*

For additional examples of rules using IP packet header information, see *Additional rule examples*, on page 4-65.

Port numbers

BIG-IP includes rule variables that you can use to select a pool based on the port number of the client or server. These variables are **client_port** and **server_port**.

To configure a rule to select a pool based on a port number, use the syntax shown in the example in Figure 4.21.

```
rule my_rule {
    if (client_port > 1000) {
        use (slow_pool)
     }
      else {
         use (fast_pool)
       }
}
```

Figure 4.21 *A rule based on a TCP or UDP port number*

IP protocol numbers

BIG-IP includes a rule variable, **ip_protocol**, that you can use to select a pool based on an IP protocol number.

To configure a rule to select a pool based on an IP protocol number, use the syntax shown in the example in Figure 4.22.

```
rule my_rule {
    if (ip_protocol == 6) {
        use (tcp_pool)
     }
      else {
         use (slow_pool)
       }
}
```

Figure 4.22 *A rule based on an IP protocol number*

Quality of Service (QoS) level

The ***Quality of Service (QoS)*** standard is a means by which network equipment can identify and treat traffic differently based on an identifier. As traffic enters the site, the BIG-IP can apply a rule that sends the traffic to different pools of servers based on the QoS level within a packet.

To configure a rule to select a pool based on the QoS level of a packet, you can use the **link_qos** rule variable, as shown in the example in Figure 4.23.

```
rule my_rule {
  if (link_qos > 2) {
    use (fast_pool)
} else {
    use (slow_pool)
}
}
```

Figure 4.23 *A rule based on a Quality of Service (QoS) level*

For information on setting QoS values on packets based on the pool selected for that packet, see *Quality of Service (QoS) level*, on page 4-44.

IP Type-Of-Service (ToS) level

The ***Type of Service (ToS)*** standard is a means by which network equipment can identify and treat traffic differently based on an identifier. As traffic enters the site, the BIG-IP can apply a rule that sends the traffic to different pools of servers based on the ToS level within a packet.

The variable that you use to set the ToS level on a packet is **ip_tos**. This variable is sometimes referred to as the **DiffServ** variable.

To configure a rule to select a pool based on the ToS level of a packet, you can use the **ip_tos** rule variable, as shown in the example in Figure 4.24.

```
rule my_rule {
if (ip_tos == 16) {
use (telnet_pool)
}
else {
use (slow_pool)
}
}
```

Figure 4.24 *A rule based on a Type of Service (ToS) level*

For information on setting ToS values on packets based on the pool selected for that packet, see *Type of Service (ToS) level*, on page 4-44.

Pool selection based on **one of** operator

BIG-IP includes a rule operator that you can use to select a pool based on whether the variable being used in the rule represents a member of a specific class.

Example

For example, prior to the availability of the **one of** operator, a rule that was intended to send incoming AOL connections to the pool **aol_pool** was written as shown in Figure 4.25, where multiple values for the **client_addr** variable had to be individually specified.

```
rule my_rule {
 if ( client_addr equals 152.163.128.0 netmask 255.255.128.0
       or client_addr equals 195.93.0.0 netmask 255.255.254.0
       or client_addr equals 205.188.128.0 netmask
             255.255.128.0 ) {
       use (aol_pool)
     }
      else {
         use (all_pool)
       }
}
```

Figure 4.25 *Example of a rule without the **one of** operator*

Using the **one of** operator instead, you can cause BIG-IP to load balance all incoming AOL connections to the pool **aol_pool,** if the value of the **client_addr** variable is a member of the class AOL. Figure 4.26 shows this type of rule. In this case, the **one of** operator indicates that the variable **client_addr** is actually a list of values (that is, a class).

```
rule my_rule {
    if (client_addr equals one of aol) {
        use (aol_pool)
     }
      else {
         use (all_pool)
       }
}
```

Figure 4.26 *A rule based on the* ***one of*** *operator*

Note that an expression such as **client_addr equals one of aol** is true if the expression is true with at least one specific value in the class.

For another example of a rule using the **one of** operator, see *Additional rule examples*, on page 4-65.

Class types

The **one of** operator for rules supports three specific types of classes. They are:

- Strings - The following command creates a string class:

  b class my_class { \".gif\" }

Note: *This example shows the use of escape characters for the quotation marks.*

Figure 4.27 shows the resulting string type of class.

```
class images {
  ".gif"
}
```

Figure 4.27 *An example of a string type of class*

- Numerics - The following command creates a numeric type of class:

  b class my_protos { 27 38 93 }

Figure 4.28 shows the resulting numeric type of class:

```
class my_protos {
    27
    38
    93
}
```

Figure 4.28 *An example of a numeric type of class*

- IP addresses - The following command creates a class containing IP addresses:

  `b class my_ntwk { network 10.2.2.0 mask 255.255.255.0 }`

Figure 4.29 shows the resulting IP address type of class:

```
class my_netwk {
  network 10.2.2.0 mask 255.255.255.0
}
```

Figure 4.29 *An example of an IP address type of class*

◆ Note

The size of a class is limited by system resources only.

Predefined lists

In addition to the **one of** operator, the BIG-IP includes a number of predefined lists for you to use with this operator. They are:

- AOL Network
- Image Extensions
- Non-routable addresses

These lists are located in the file **/etc/default_classes.txt**. When the **bigpipe load** command is issued, the lists are loaded. Unless modified by a user, these lists are not saved to the file **bigip.conf**.

To view classes

To view classes, including the default classes, use the following command.

`bigpipe class show`

Pool selection based on HTTP header data

A rule can contain a cache statement that selects a pool based on HTTP header data. A cache statement returns either the origin pool, the hot pool, or the cache pool. When the cache pool is selected, it is accompanied by the

indicated node address and port. When a rule returns both a pool and a node, the BIG-IP does not do any additional load balancing or persistence processing.

Figure 4.30 shows an example of a rule containing a cache statement.

```
rule my_rule {
    if ( http_host starts_with "dogfood" ) {
        cache ( http_uri ends_with "html" or http_uri ends_with "gif" ) {
        origin_pool origin_server
        cache_pool cache_servers
        hot_pool cache_servers
        hot_threshold 100
        cool_threshold 10
        hit_period 60
        content_hash_size 1024
        }
    }
    else {
        use ( cathost named_servers )
    }
}
```

Figure 4.30 *An example of a cache load balancing rule*

For a complete list of cache statement syntax, see *Configuring a remote origin server*, on page 4-64.

Rule-based HTTP redirection

In addition to configuring a rule to select a specific pool, you can also configure a rule to redirect an HTTP request to a specific location, using the **redirect to** operator and a set of format strings included in the BIG-IP. The location can be either a host name, port number, or URI. The format strings are: **%h**, **%p**, and **%u**. These format strings can be used within a redirection string to indicate which parts of the string (host name, port number, and URI path) do not indicate a redirection.

For example, the string **https://%h:443/%u** specifies that the HTTP request is to be redirected to a different protocol (**https** instead of the standard **http**) and a different port number (**443**). The host name and the URI path remain the same, indicated by the **%h** and **%u** format strings.

Figure 4.31 shows a rule that is configured to redirect an HTTP request.

```
rule my_rule {
    if (http_uri ends_with "baz") {
        redirect to "https://%h:8080/%u/"
     }
      else {
         use (web_pool)
       }
}
```

Figure 4.31 *A rule based on HTTP redirection*

The preceding rule applies the format string to the URL. In this case, the format string sets the protocol to **https**, strips the requested port number (if any), and changes it to **8080**, and applies a trailing slash (/) to the end of the URI, if the URI ends with the string **baz**.

◆ Note

*The **%u** format string strips the first character of the URI path. This is usually a slash (/), and this modification is done purely for aesthetic reasons. Thus when describing a URL, the string **http://%h/%u** is used instead of **http://%h%u**.*

For more information on HTTP redirection and format strings, see *HTTP redirection*, on page 4-37.

Rule statements

A rule consists of statements. Rules support the following types of statements:

- An **if** statement asks a true or false question and, depending on the answer, takes some action.
- A **discard** statement discards the request. This statement must be conditionally associated with an **if** statement.
- A **use** statement uses a selected pool for load balancing. This statement must be conditionally associated with an **if** statement.
- A **cache** statement uses a selected pool for load balancing. This statement can be conditionally associated with an **if** statement.
- A **redirect** statement sends traffic to a specific destination, rather than to a pool for load balancing.

The primary possible statements expressed in command line syntax are:

```
if (<question>) {<statement>} [else {<statement>}]
discard
use ( <pool_name> )
cache ( <expressions> )
redirect ( <redirect URL> )
```

For detailed syntax of these rules statements, see *To define a rule from the command line*, on page 4-63.

Questions (expressions)

A question or expression is asked by an **if** statement and has a true or false answer. A question or expression has two parts: a predicate (**operator**), and one or two subjects (**operands**).

There are two types of subjects (operands); some subjects change and some subjects stay the same.

- Changing subjects are called ***variable operands***.

- Subjects that stay the same are called ***constant operands***.

A question, or ***expression***, asks questions about variable operands by comparing their current value to constant operands with relational operators.

Constant operands

Possible constant operands are:

- IP protocol constants, for example:
  UDP or **TCP**
- IP addresses expressed in masked dot notation, for example:
  `206.0.0.0 netmask 255.0.0.0`
- Strings of ASCII characters, for example:
  `pictures/bigip.gif`
- Regular expression strings

Variable operands (variables)

Since variable operands change their value, they need to be referred to by a constant descriptive name. The variables available depend on the context in which the rule containing them is evaluated. Possible variable operands are:

- IP packet header variables, such as:
  - **client_addr**. Used by a client to represent a source IP address. This variable is replaced with an unmasked IP address.
  - **server_addr**. Used to represent a destination IP address. This variable is replaced with an unmasked IP address. The **server_addr** variable is used to represent the destination address of the packet. This variable is useful when load balancing traffic to a wildcard virtual server.
  - **client_port**. Used to represent a client port number.
  - **server_port**. Used to represent a server port number.
  - **ip_protocol**. Used to represent an IP protocol. This variable is replaced with a numeric value representing an IP protocol such as TCP, UDP, or IPSEC.
  - **link_qos**. Used to represent the Quality of Service (QoS) level.
  - **ip_tos**. Used to represent that Type of Service (ToS) level.
- HTTP request strings

  All HTTP request string variables are replaced with string literals. HTTP request variables are referred to in command line syntax by a predefined set of names. Internally, an HTTP request variable points to a method for extracting the desired string from the current HTTP request header data. Before an HTTP request variable is used in a relational expression, it is replaced with the extracted string.

  The evaluation of a rule is triggered by the arrival of a packet. Therefore, variables in the rule may refer to features of the triggering packet. In the

case of a rule containing questions about an HTTP request, the rule is evaluated in the context of the triggering TCP SYN packet until the first HTTP request question is encountered. After the proxy, the rule continues evaluation in the context of the HTTP request packet, and variables may refer to this packet. Before a variable is compared to the constant in a relational expression, it is replaced with its current value.

The allowed variable names are:

- **http_method**
  The **http_method** is the action of the HTTP request. Common values are **GET** or **POST**.
- **http_uri**
  The **http_uri** is the URL, but does not include the protocol and the fully qualified domain name (FQDN). For example, if the URL is **http://www.url.com/buy.asp**, then the URI is **/buy.asp**.
- **http_version**
  The **http_version** is the HTTP protocol version string. Possible values are "**HTTP/1.0**" or "**HTTP/1.1**".
- **http_host**
  The **http_host** is the value in the Host: header of the HTTP request. It indicates the actual FQDN that the client requested. Possible values are a FQDN or a host IP address in dot notation.
- **http_cookie <cookie name>**
  The HTTP cookie header is the value in the Cookie: header for the specified cookie name. An HTTP cookie header line can contain one or more cookie name value pairs. The **http_cookie <cookie name>** variable evaluates to the value of the cookie with the name **<cookie name>**. For example, given a request with the following cookie header line:

  Cookie: green-cookie=4; blue-cookie=horses

  The variable **http_cookie blue-cookie** evaluates to the string **horses**. The variable **http_cookie green-cookie** evaluates to the string **4**.
- **http_header <header_tag_string>**
  The variable **http_header** evaluates the string following an HTTP header tag that you specify. For example, you can specify the **http_host** variable with the **http_header** variable. In a rule specification, if you wanted to load balance based on the host name **andrew,** the rule statement might look as follows:

```
if ( http_header "Host" starts_with "andrew" )  { use ( andrew_pool ) } else { use (
    main_pool ) }
```

Operators

In a rule, ***relational operators*** compare two operands to form relational expressions. Possible relational operators and expressions are described in Table 4.10.

Expression	Relational Operator
Are two IP addresses equal?	**<address> equals <address>**
Do a string and a regular expression match?	**<variable_operand> matches_regex <regular_expression>**
Are two strings identical?	**<string> equals <string>**
Is the second string a suffix of the first string?	**<variable_operand> ends_with <string>**
Is the second string a prefix of the first string?	**<variable_operand> starts_with <string>**
Does the first string contain the second string?	**<variable_operand> contains <literal_string>**
Is the first string a member of the second string?	**<variable_operand> equals one of <class>**

Table 4.10 *The relational operators*

In a rule, ***logical operators*** modify an expression or connect two expressions together to form a logical expression. Possible logical operators and expressions are described in Table 4.11.

Expression	Logical Operator
Is the expression not true?	**not <expression>**
Are both expressions true?	**<expression> and <expression>**
Is either expression true?	**<expression> or <expression>**

Table 4.11 *The logical operators*

Cache statements

A **cache** statement may be either the only statement in a rule or it may be nested within an **if** statement. Rules with **cache** statements are used to select pools based on HTTP header data. Table 4.12 describes the **cache** statement syntax.

Rule Syntax	Description
expression	A Boolean expression setting the condition or conditions under which the rule applies.
origin_pool <pool_name>	This required attribute specifies a pool of servers with all the content to which requests are load balanced when the requested content is not cacheable or when all the cache servers are unavailable or when you use a BIG-IP to redirect a miss request from a cache.
cache_pool <pool_name>	This required attribute specifies a pool of cache servers to which requests are directed to optimize cache performance.
hot_pool <pool_name>	This optional attribute specifies a pool of servers that contain content to which requests are load balanced when the requested content is frequently requested (hot). If you specify any of the following attributes in this table, the **hot_pool** attribute is required.
hot_threshold <hit_rate>	This optional attribute specifies the minimum number of requests for content that cause the content to change from cool to hot at the end of the period (**hit_period**).
cool_threshold <hit_rate>	This optional attribute specifies the maximum number of requests for specified content that cause the content to change from hot to cool at the end of the period.
hit_period <seconds>	This optional attribute specifies the period in seconds over which to count requests for particular content before deciding whether to change the hot or cool state of the content.
content_hash_size <sets_in_content_hash>	This optional attribute specifies the number subsets into which the content is divided when calculating whether content is hot or cool. The requests for all content in the same subset are summed and a single hot or cool state is assigned to each subset. This attribute should be within the same order of magnitude as the actual number of requests possible. For example, if the entire site is composed of 500,000 pieces of content, a **content_hash_size** of 100,000 would be typical.

Table 4.12 *Description of **cache** statement syntax*

For a description of pool selection based on HTTP header data, see *Pool selection based on HTTP header data*, on page 4-56.

Configuring rules

You can create rules from the command line or by using the Configuration utility. Each of these methods is described in this section.

To define a rule using the Configuration utility

1. In the navigation pane, click **Rules**.
   This opens the Rules screen.

2. Click the **Add** button.
   The Add Rule screen opens.
3. In the Add Rule screen, fill in the fields to add a rule.
   You can type in the rule as an unbroken line, or you can use the Enter key to add line breaks.
4. Click **Done**.

To define a rule from the command line

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

```
b rule <rule_name>  '{ <if statement> | <cache_statement }'
```

Table 4.13 contains descriptions of all the elements you can use to create rules.

Element	Description
rule definition	`rule <rule_name { <if_statement> | <discard_statement> | <use_statement> | <<cache_statement> | 41 <redirect_statement>}`
if statement	`if ( <expression> ) { <statement> } [ { else <statement> } ] [ { else if <statement> } ]`
discard statement	`discard |`
use statement	`use ( <pool_name> )`
cache statement	`cache ( <expression> ) { origin_pool <pool_name> cache_pool <pool_name> [ hot_pool <pool_name> ] [ hot_threshold <hit_rate> ] [ cool_threshold <hit_rate> ] [ hit_period <seconds> ][ content_hash_size <sets_in_content_hash> ] }`
redirect statement	`redirect ( <redirect URL> )`
literal	`<regex_literal>` `<string_literal>` `<address_literal>`
regular expression literal	Is a string of 1 to 63 characters enclosed in quotes that may contain regular expressions
string literal	Is a string of 1 to 63 characters enclosed in quotes
address literal	`<dot_notation_longword> [netmask <dot_notation_longword>]`
Dot notation longword	`<0-255>.<0-255>.<0-255>.<0-255>`

Table 4.13 *Elements for rules construction*

Element	Description
variable	**http_method** **http_header <header tag>** **http_version** **http_uri** **http_host** **http_cookie <cookie_name>** **link_qos** **ip_tos** **client_addr** **server_addr** **client_port** **server_port** **ip_protocol**
binary operator	**or** **and** **contains** **matches** **equals** **starts_with** **ends_with** **matches_regex** **one of** **redirect to**

Table 4.13 *Elements for rules construction*

Configuring a remote origin server

To ensure that a remote origin server or cache server responds to the BIG-IP rather than to the original cache server that generated the missed request, the BIG-IP also translates the source of the missed request to the translated address and port of the associated SNAT connection.

In order to enable these scenarios, you must:

- Create a SNAT for each cache server.
- Create a SNAT auto-mapping for bounceback.

Configuring a SNAT for each origin server

The SNAT translates the address of a packet from the cache server to the address you specify. For more information about SNATs, see *SNATs*, on page 4-122.

Creating a SNAT automap for bounceback

You must now configure a second SNAT mapping, in this case with the SNAT automap feature, so that when requests are directed to the origin server, the server will reply through the BIG-IP and not directly to the client. (If the BIG-IP replied directly to the client, the next request would then go directly to the origin server, removing the BIG-IP from the loop.) For more information about SNATs, see *SNATs*, on page 4-122.

Additional rule examples

This section contains additional examples of rules including:

- Cookie rule
- Language rule
- Cache rule
- AOL rule
- Protocol specific rule

Cookie rule

Figure 4.32 shows a cookie rule that load balances based on the user ID that contains the word **VIRTUAL**.

```
if ( exists http_cookie "user-id" and
     http_cookie "user-id" contains "VIRTUAL" ) {
   use ( virtual_pool )
}
else {
   use ( other_pool )
}
```

Figure 4.32 *Cookie rule example*

Language rule

Figure 4.33 shows a rule that load balances based on the language requested by the browser.

```
if ( exists http_header "Accept-Language" ) {
   if ( http_header "Accept-Language" equals "fr" ) {
      use ( french_pool )
   }
   else if ( http_header "Accept-Language" equals "sp" ) {
         use (spanish_pool )
   }
   else {
   use ( english_pool )
}
```

Figure 4.33 *Sample rule that load balances based on the language requested by the browser*

Cache rule

Figure 4.34 shows an example of a rule that you can use to send cache content, such as **.gifs**, to a specific pool.

```
if ( http_uri ends_with "gif" or
     http_uri ends_with "html" ) {
   use ( cache_pool )
}
else {
   use ( server_pool )
}
```

Figure 4.34 *An example of a cache rule*

AOL rule

Figure 4.35 is an example of a rule that you can use to load balance incoming AOL connections.

```
port 80 443 enable
pool aol_pool {
   min_active_members 1
   member 12.0.0.31:80 priority 4
   member 12.0.0.32:80 priority 3
   member 12.0.0.33:80 priority 2
   member 12.0.0.3:80 priority 1
}
pool other_pool {
   member 12.0.0.31:80
   member 12.0.0.32:80
   member 12.0.0.33:80
   member 12.0.0.3:80
}
pool aol_pool_https {
   min_active_members 1
   member 12.0.0.31:443 priority 4
   member 12.0.0.32:443 priority 3
   member 12.0.0.33:443 priority 2
   member 12.0.0.3:443 priority 1
}
pool other_pool_https{
   member 12.0.0.31:443
   member 12.0.0.32:443
   member 12.0.0.33:443
   member 12.0.0.3:443
}
rule aol_rule {
   if (client_addr equals one of aol) {
      use ( aol_pool )
   }
   else {
      use ( other_pool)
   }
}
```

Figure 4.35 *An example of an AOL rule*

```
rule aol_rule_https {
   if (    client_addr equals 152.163.128.0 netmask 255.255.128.0
        or client_addr equals 195.93.0.0    netmask 255.255.254.0
        or client_addr equals 205.188.128.0 netmask 255.255.128.0 ) {
      use ( aol_pool_https )
   }
   else {
      use ( other_pool_https)
   }
}
virtual 15.0.140.1:80  { use rule aol_rule }
virtual 15.0.140.1:443 { use rule aol_rule_https special ssl 30 }
```

Figure 4.35 *An example of an AOL rule*

Rule using the ip_protocol variable

Figure 4.36 shows a rule that uses the **ip_protocol** variable.

```
rule myrule {
   if ( ip_protocol == 37 ) {
      use ( bootp_pool )
   } else if ( ip_protocol == 22 ){
      use ( ipsec_pool )
   }
    else {
       use ( slow_pool )
   }
  }
```

Figure 4.36 *An example of an IP protocol rule*

Rule using IP address and port variables

Figure 4.37 shows a rule that uses the **server_addr** and **server_port** rule variables.

```
rule myrule {
   if ( server_addr  equals 10.0.0.0 netmask 255.255.0.0 ) {
      use ( fast_pool )
   } else if ( server_port equals 80 ){
      use ( fast_pool )
   }
    else {
       use ( slow_pool )
   }
  }
```

Figure 4.37 *An example of an IP protocol rule*

Rule using the **one of** operator

A good use of the **one of** operator in a rule is when you have a class such as that shown in Figure 4.38.

```
class images {
  ".gif"
  ".jpg"
  ".bmp"
}
```

Figure 4.38 *An example of a class*

Given the above class, you could create a rule that uses the **one of** operator to select a pool based on whether the value of the variable **http_uri** ends with a member of the class **images**. Figure 4.39 shows this rule.

```
rule myrule {
   if ( http_uri ends_with one of images ) {
     use ( image_pool )
   }
   else {
      use ( dynamic_pool )
   }
```

Figure 4.39 *Example of a rule using the one of operator*

Rules based on HTTP header insertion

You can create rules based on headers that an SSL Accelerator proxy has inserted into HTTP requests. For examples of these types of rules, see *Inserting headers into HTTP requests*, on page 4-95.

Virtual servers

A virtual server with its virtual address is the visible, routable entity through which nodes in a load balancing pool are made available to a client, either directly or indirectly through a rule. (The exception is the forwarding virtual server, which simply forwards traffic and has no associated pools.)

You must configure a pool of servers before you can create a virtual server that references the pool. Before you configure virtual servers, you need to know:

- Which virtual server type meets your needs
- Whether you need to activate optional virtual server properties

Once you know which virtual server options are useful in your network, you can define any one of the four types of virtual servers.

Virtual server types

You can configure various types of virtual servers, depending on your needs. Table 4.14 shows the types of virtual servers that you can create.

Virtual Server Type	Description
Standard virtual server	A standard virtual server is a virtual server with a full IP address. For example: **192.168.200.30:http**
Wildcard virtual server	There are two types of wildcard servers: A ***port-specific wildcard virtual server*** is a virtual server with a port specified. A port-specific wildcard virtual server is used to accept all traffic for a specific service. A ***default wildcard virtual server*** is a wildcard virtual server with the service **0**, *****, or **any**. A default wildcard server acts like a default router, accepting all traffic that does not match a standard, network, or port-specific wildcard server.
Network virtual server	A network virtual server is a virtual server with a network IP address, allowing it to handle a whole range of addresses in a network. For example: **192.168.200.0:http**
Forwarding virtual server	A forwarding virtual server is a virtual server without a pool that simply forwards traffic to the destination node.

Table 4.14 *Virtual server types*

Standard virtual servers

A s***tandard virtual server*** represents a specific site, such as an Internet web site or an FTP site, and it load balances content servers that are members of a pool. The IP address that you use for a standard virtual server should match the IP address that DNS associates with the site's domain name.

◆ Note

*If you are using a 3-DNS Controller in conjunction with the BIG-IP, the 3-DNS Controller uses the IP address associated with the registered domain name in its own configuration. For details, refer to the **3-DNS Administrator Guide**.*

To define a standard virtual server using the Configuration utility

1. In the navigation pane, click **Virtual Servers**.
2. Click the **Add** button.
   The Add Virtual Server screen opens.
3. In the **Address** box, type the virtual server's IP address or host name.
4. In the **Port** box, either type a port number or select a service name from the list.
5. In the Select Physical Resources screen, click the **Pool** button.
   If you want to assign a load balancing rule to the virtual server, click **Rule** and select a rule you have configured.
6. In the **Pool** list, select the pool you want to apply to the virtual server.
7. Click the **Apply** button.

To define a standard virtual server from the command line

Type the **bigpipe virtual** command as shown below. Also, remember that you can use host names in place of IP addresses, and that you can use standard service names in place of port numbers.

```
b virtual <virt_ip>:<service> use pool <pool_name>
```

For example, the following command defines a virtual server that maps to the pool **my_pool**:

```
b virtual 192.200.100.25:80 use pool my_pool
```

◆ Note

If a virtual server is to have the the same IP address as a node in an associated VLAN, you must perform some additional configuration tasks. These tasks consist of: creating a VLAN group that includes the VLAN in which the node resides, assigning self-IP addresses to the VLAN group, and disabling the virtual server on the relevant VLAN. For information on

creating VLAN groups and assigning self IP addresses to them, see Chapter 3, Creating VLAN groups, on page 3-14. For information on disabling a virtual server for a specific VLAN, see Enabling or disabling a virtual server, on page 4-84.

Wildcard virtual servers

Wildcard virtual servers are a special type of virtual server designed to manage network traffic for transparent network devices, such as transparent firewalls, routers, proxy servers, or cache servers. A wildcard virtual server manages network traffic that has a destination IP address unknown to the BIG-IP. A standard virtual server typically represents a specific site, such as an Internet web site, and its IP address matches the IP address that DNS associates with the site's domain name. When the BIG-IP receives a connection request for that site, the BIG-IP recognizes that the client's destination IP address matches the IP address of the virtual server, and it subsequently forwards the client to one of the content servers that the virtual server load balances.

However, when you are load balancing transparent nodes, a client's destination IP address is going to seem random. The client is connecting to an IP address on the other side of the firewall, router, or proxy server. In this situation, the BIG-IP cannot match the client's destination IP address to a virtual server IP address. Wildcard virtual servers resolve this problem by not translating the incoming IP address at the virtual server level on the BIG-IP. For example, when the BIG-IP does not find a specific virtual server match for a client's destination IP address, it matches the client's destination IP address to a wildcard virtual server. The BIG-IP then forwards the client's packet to one of the firewalls or routers that the wildcard virtual server load balances, which in turn forwards the client's packet to the actual destination IP address.

Default vs. port-specific wildcard servers

When you configure wildcard virtual servers and the nodes that they load balance, you can use a wildcard port (port **0**) in place of a real port number or service name. A wildcard port handles any and all types of network services.

A wildcard virtual server that uses port **0** is referred to as a ***default wildcard virtual server***, and it handles traffic for all services. A ***port-specific wildcard virtual server*** handles traffic only for a particular service, and you define it using a service name or a port number. If you use both a default wildcard virtual server and port-specific wildcard virtual servers, any traffic that does not match either a standard virtual server or one of the port-specific wildcard virtual servers is handled by the default wildcard virtual server.

By default, a default wildcard virtual server is enabled for all VLANs. However, you can specifically disable any VLANs that you do not want the default wildcard virtual server to support. Disabling VLANs for the default wildcard virtual server is done by creating a VLAN disabled list. Note that a

VLAN disabled list applies to default wildcard virtual servers only. You cannot create a VLAN disabled list for a wildcard virtual server that is associated with one VLAN only.

You can use port-specific wildcard virtual servers for tracking statistics for a particular type of network traffic, or for routing outgoing traffic, such as HTTP traffic, directly to a cache server rather than a firewall or router.

We recommend that when you define transparent nodes that need to handle more than one type of service, such as a firewall or a router, you specify an actual port for the node and turn off port translation for the virtual server.

For the procedure to create a default wildcard server, see *To create a default wildcard virtual server using the Configuration utility*, on page 4-73.

Creating wildcard virtual servers

Creating a wildcard virtual server requires three steps. First, you must create a pool that contains the addresses of the transparent devices. Next, you must create the wildcard virtual server. Then you must turn port translation off for the virtual server. The following sections describe these steps, followed by the procedure for creating a default wildcard server.

To create a pool of transparent devices using the Configuration utility

To create a pool of transparent devices, use the Add Pool wizard, available from the Pools screen. For more information, see *To create a pool using the Configuration utility*, on page 4-3.

To create a wildcard virtual server using the Configuration utility

1. In the navigation pane, click **Virtual Servers**.
2. Click the **Add** button.
   The Add Virtual Server screen opens.
3. In the **Address** box, type the wildcard IP address **0.0.0.0**.
4. In the **Port** box, type a port number, or select a service name from the list. Note that port **0** defines a wildcard virtual server that handles all types of services. If you specify a port number, you create a port-specific wildcard virtual server. The wildcard virtual server handles traffic only for the port specified. For more information, see *Default vs. port-specific wildcard servers*, on page 4-71.
5. In Resources, click the **Pool** button.
6. In the Pool list, select the pool you want to apply to the virtual server.
7. Click the **Apply** button.

To turn off port translation for a wildcard virtual server using the Configuration utility

After you define the wildcard virtual server with a wildcard port, you must disable port translation for the virtual server.

1. In the navigation pane, click **Virtual Servers**.
   The Virtual Servers screen opens.
2. In the virtual server list, click the virtual server for which you want to turn off port translation.
   The Virtual Server Properties screen opens.
3. In the Enable Translation section, clear the **Port** box.
4. Click the **Apply** button.

To create a wildcard virtual server from the command line

1. Create the pool of transparent devices, using the **bigpipe pool** command. For example, you can create a pool of transparent devices called **transparent_pool** that uses the Round Robin load balancing mode:

   ```
   b pool transparent_pool { \
   member 10.10.10.101:80 member 10.10.10.102:80 \
   member 10.10.10.103:80 }
   ```

2. Create a wildcard virtual server that maps to the pool **transparent_pool**. Because the members are firewalls and need to handle a variety of services, the virtual server is defined using port **0** (or ***** or **any**). You can specify any valid non-zero port for the node port and then turn off port translation for that port. In this example, service checks ping port **80**. For example:

   ```
   b virtual 0.0.0.0:0 use pool transparent_pool
   ```

3. Turn off port translation for the port in the virtual server definition. In the following example, port **80** is used for service checking. If you do not turn off port translation, all incoming traffic is translated to port **80**.

   ```
   b virtual 0.0.0.0:0 translate port disable
   ```

To create a default wildcard virtual server using the Configuration utility

1. In the Navigation pane, select **Virtual Servers**.
   The Virtual Servers screen displays.
2. Click the **Add** button.
3. In the **Address** field, type the IP address **0.0.0.0**.
4. Click **Next**.
5. From the **VLAN** box, select **all**.
6. Click **Done**.

To create a default wildcard virtual server from the command line

To create a default wildcard virtual server from the command line, use the **bigpipe virtual** command with the following syntax:

```
b virtual *:* use pool <pool_name>
```

Creating multiple wildcard servers

In previous releases, BIG-IP supported one wildcard virtual server only, designated by the IP address **0.0.0.0**. With this release, you can define multiple wildcard virtual servers, all running simultaneously. Each wildcard virtual server must be assigned to an individual VLAN, and therefore handles packets for that VLAN only.

To create multiple wildcard virtual servers, you can use either the Configuration utility or the **bigpipe virtual** command.

To create multiple wildcard virtual servers using the Configuration utility

To create multiple wildcard virtual servers using the Configuration utility, use the following procedure:

1. In the Navigation pane, select **Virtual Servers**.
   The Virtual Servers screen displays.
2. Click the **Add** button.
3. In the **Address** field, type the IP address **0.0.0.0**.
4. In the **Service** field, type the name of a service or select a service from the list box.
5. Click **Next**.
6. From the VLAN box, choose a VLAN name. Selecting **all** creates a default wildcard virtual server.
7. Click **Next**.
8. Continue configuring all properties for the wildcard virtual server. Note that on the Configure Basic Properties screen, if you are creating a default wildcard virtual server, you can disable any VLANs associated with that wildcard virtual server.
9. Click **Done**.

Repeat for each wildcard virtual server that you want to create.

To create multiple wildcard virtual servers from the command line

To create a separate wildcard virtual server per VLAN from the command line, use the following command-line syntax:

```
b virtual <vlan_name> use pool <pool_name>
```

For example, the following commands define two wildcard virtual servers, the first for VLAN internal, and the second for VLAN external:

```
b virtual internal use pool my_pool
b virtual external use pool my_pool
```

Network virtual servers

You can configure a ***network virtual server*** to handle a whole network range, instead of just one IP address, or all IP addresses (a wildcard virtual server). For example, the virtual server in Figure 4.40 handles all traffic addresses in the 192.168.1.0 network.

```
bigpipe virtual 192.168.1.0:0 {
      netmask 255.255.255.0 use pool ingress_firewalls
}
```

Figure 4.40 *A sample network virtual server*

A network virtual server is a virtual server that has no bits set in the host portion of the IP address. The example above directs all traffic destined to the subnet **192.168.1.0/24** through the BIG-IP to the **ingress_firewalls** pool.

The netmask of a network virtual server establishes which portion of the address is actually the network of a network virtual server. By default, this is the netmask of the self IP address. In the example, the network mask of **255.255.255.0** states that the network portion of the address is **192.168.1**, which in this case is obvious because only the last octet has a zero value.

A less obvious case would be the network virtual server **10.0.0.0:0**. Here, the zero in the second octet is ambiguous: it could be a wildcard or it could be a literal zero. If it is a wildcard, this would be established by a netmask of **255.0.0.0**. If it is a literal zero, this would be established by a netmask of **255.255.0.0**.

Another way you can use this feature is to create a catch-all web server for an entire subnet. For example, you could create the following network virtual server, shown in Figure 4.41.

```
bigpipe virtual 192.168.1.0:http {
    netmask 255.255.255.0 broadcast 192.168.1.255
    use pool default_webservers
}
```

Figure 4.41 *A catch-all web server configuration.*

This configuration directs a web connection destined to any address within the subnet **192.168.1.0/24** to the **default_webservers** pool.

Forwarding virtual servers

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

To configure forwarding virtual servers using the Configuration utility

1. In the navigation pane, click **Virtual Servers**.
   The Virtual Servers screen opens.
2. Click the **Add** button.
   The Add Virtual Server screen opens.
3. Type the virtual server attributes, including the address and port number.
4. Under **Configure Basic Properties**, clear the check from **Enable Arp**.
5. On the Select Physical Resources screen, click the **Forwarding** button.
6. Click the **Apply** button.

To configure a forwarding virtual server from the command line

Use the following syntax to configure forwarding virtual servers:

```
b virtual <virt_ip>:<service> forward
b virtual <virt_ip>:<service> arp disable
```

For example, to allow only one service in:

```
b virtual 206.32.11.6:80 forward
b virtual <virt_ip>:<service> arp disable
```

Use the following command to allow only one server in:

```
b virtual 206.32.11.5:0 forward
b virtual <virt_ip>:<service> arp disable
```

To forward all traffic, use the following command:

```
b virtual 0.0.0.0:0 forward
```

In some of the configurations described here, you need to set up a wildcard virtual server on one side of the BIG-IP to load balance connections across transparent devices. You can create another wildcard virtual server on the

other side of the BIG-IP to forward packets to virtual servers receiving connections from the transparent devices and forwarding them to their destination.

◆ Tip

*If you do not want BIG-IP to load balance your traffic but do want to take advantage of certain pool attributes, you can instead use a feature called a **forwarding pool**. For more information on forwarding pools, see Forwarding pools, on page 4-47.*

◆ Note

If a forwarding virtual server is to have the the same IP address as a node in an associated VLAN, you must perform some additional configuration tasks. These tasks consist of: creating a VLAN group that includes the VLAN in which the node resides, assigning self-IP addresses to the VLAN group, and disabling the virtual server on the relevant VLAN. For information on creating VLAN groups and assigning self IP addresses to them, see Chapter 3, Creating VLAN groups, on page 3-14. For information on disabling a virtual server for a specific VLAN, see Enabling or disabling a virtual server, on page 4-84.

Virtual server options

For each type of virtual server, you can configure several options. These options are shown in 4.15.

Option	Description
Mirroring information	You can use mirroring to maintain the same state information in the standby unit that is in the active unit, allowing transactions such as FTP file transfers to continue as though uninterrupted.
Netmask and broadcast	You can override the default netmask and broadcast for a network virtual address.
Connection limits	You can set a concurrent connection limit on one or more virtual servers.
Translation properties	You can turn port translation **off** for a virtual server if you want to use the virtual server to load balance connections to any service.
Dynamic connection rebinding	You can cause any connections that were made to a node address or service to be redirected to another node, if the original node transitions to a **down** state.
Last hop pools	You can direct reply traffic to the last hop router using a last hop pool. This overrides the **auto_lasthop** setting.

***Table 4.15** Virtual server configuration options*

Option	Description
Rules	You can configure a virtual server to reference a rule. Rules are primarily used for selecting pools during load balancing.
Software acceleration	You can speed up packet flow for TCP connections when the packets are not fragmented.

Table 4.15 *Virtual server configuration options*

Mirroring virtual server state

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

◆ Note

Mirroring slows BIG-IP performance and is primarily useful for long-lived services like FTP and Telnet. Mirroring is not useful for short-lived connections like HTTP.

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

To control mirroring for a virtual server

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

```
b virtual <virt addr>:<service> mirror [conn] enable|disable
```

To mirror connection information for the virtual server

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

```
b virtual <virt addr>:<service> mirror [conn] show
```

If you do not specify **conn** for connection information, the BIG-IP assumes that you want to display that information.

◆ Note

If you set up mirroring on a virtual server that supports FTP connections, you need to mirror the control port virtual server, and the data port virtual server.

The following example shows the two commands used to enable mirroring for virtual server **v1** on the FTP control and data ports:

```
b virtual v1:21 mirror conn enable
b virtual v1:20 mirror conn enable
```

Displaying information about virtual servers

You can display information about all virtual servers in your configuration, or you can display information about one or more specific virtual servers.

To display information about all virtual servers in your configuration

Use the following syntax to display information about all virtual servers included in the configuration:

```
b virtual show
```

To display information about one or more virtual servers in your configuration

Use the following syntax to display information about one or more virtual servers included in the configuration:

```
b virtual <virt_ip>:<service> [...<virt_ip>:<service>] show
```

The command displays information such as the nodes associated with each virtual server, the nodes' status, and the current, total, and maximum number of connections managed by the virtual server since the BIG-IP was last rebooted.

Setting a user-defined netmask and broadcast

The default netmask for a virtual address, and for each virtual server hosted by that virtual address, is determined by the network class of the IP address entered for the virtual server. The default broadcast is automatically determined by the BIG-IP, and it is based on the virtual address and the current netmask. You can override the default netmask and broadcast for a network virtual address only.

All virtual servers hosted by the virtual address use the netmask and broadcast of the virtual address, whether they are default values or they are user-defined values.

To set a custom netmask and broadcast

If you want to use a custom netmask and broadcast, you define both when you define the network virtual server:

```
b virtual <virt_ip>[:<service>] [vlan <vlan_name> disable | enable] [netmask <ip>]
    [broadcast <ip>] use pool <pool_name>
```

◆ Note

The BIG-IP calculates the broadcast based on the IP address and the netmask. In most cases, a user-defined broadcast address is not necessary.

Again, even when you define a custom netmask and broadcast in a specific network virtual server definition, the settings apply to all virtual servers that use the same virtual address. The following sample command shows a user-defined netmask and broadcast:

```
b virtual www.SiteOne.com:http \
netmask 255.255.0.0 \
broadcast 10.0.140.255 \
use pool my_pool
```

The **/bitmask** option shown in the following example applies network and broadcast address masks. In this example, a 24-bit bitmask sets the network mask and broadcast address for the virtual server:

```
b virtual 206.168.225.0:80/24 use pool my_pool
```

You can generate the same broadcast address by applying the **255.255.255.0** netmask. The effect of the bitmask is the same as applying the **255.255.255.0** netmask. The broadcast address is derived as **206.168.225.255** from the network mask for this virtual server.

Setting a connection limit

The default setting is to have no limit to the number of concurrent connections allowed on a virtual server.

To set a concurrent connection limit

You can set a concurrent connection limit on one or more virtual servers using the following command:

```
b virtual <virt_ip> [:<service>] \ [...<virt_ip>[:<service>]] limit <max_conn>
```

The following example shows two virtual servers set to have a concurrent connection limit of 5000 each:

```
b virtual www.SiteOne.com:http www.SiteTwo.com:ssl limit 5000
```

To turn off the connection limit

To turn off the connection limit, set the <**max conn**> variable to zero:

```
b virtual <virt_ip>[:<service>] [...<virt_ip>[:<service>] ] limit 0
```

Setting translation properties for virtual addresses and ports

Turning off port translation for a virtual server is useful if you want to use the virtual server to load balance connections to any service.

You can also configure the translation properties for a virtual server address. This option is useful when the BIG-IP is load balancing devices that have the same IP address. This is typical with the nPath routing configuration where duplicate IP addresses are configured on the loopback device of several servers.

To enable or disable port translation

Use the following syntax to enable or disable port translation for a virtual server:

```
b virtual <virt_ip>:<service> translate port enable | disable | show
```

To enable or disable address translation

Use the following syntax to enable or disable address translation for a virtual server:

```
b virtual <virt_ip>:<service> translate addr enable | disable | show
```

Setting dynamic connection rebinding

Dynamic connection rebinding is a feature for those virtual servers that are load balancing transparent devices such as firewalls or routers. Dynamic connection rebinding causes any connections that were made to a node address or service to be redirected to another node, if the original node transitions to a **down** state. In this case, all connections to the failed node that were made through the virtual server are moved to a newly-selected node from the virtual server's pool. The new node is selected using the pool's load-balancing algorithm. By default, dynamic connection rebinding is disabled.

◆ Note

This feature does not apply to virtual servers for non-transparent devices because they usually involve application state between the client and server node. This state cannot be recreated with a newly-selected node.

To enable, disable, or show the status of dynamic connection rebinding, you can use either the Configuration utility or the **bigpipe virtual** command.

To set dynamic connection rebinding using the Configuration utility

To set dynamic connection rebinding using the Configuration utility, use the following procedure.

1. In the Navigation pane, click **Virtual Servers**.
   The Virtual Servers screen opens.
2. Select the IP address for the virtual server. This displays the Properties page for that server.
3. Check the **Enable Connection Rebind** check box.
4. Click the **Apply** button.

To set dynamic connection rebinding from the command line

To manage dynamic connection rebinding using the **bigpipe virtual** command, type one of the following commands.

```
b virtual <ip>:<service> conn rebind enable
b virtual <ip>:<service> conn rebind disable
b virtual <ip>:<service> conn rebind show
```

Setting up last hop pools for virtual servers

In cases where you have more than one router sending connections to a BIG-IP, connections are automatically sent back through the same router from which they were received when the **auto_lasthop** global variable is enabled, as it is by default. If you want to exclude one or more routers from **auto-lasthop**, or if the global **auto_lasthop** is disabled for any reason (for example, you may not want it for an SSL gateway), you can use a last hop pool instead. (If **auto_lasthop** is enabled, the last hop pool takes precedence over it.)

To configure a last hop pool, you must first create a pool containing the router inside addresses. After you create the pool, use the following syntax to configure a last hop pool for a virtual server:

```
b virtual <virt_ip>:<service> lasthop pool <pool_name> | none | show
```

Configuring virtual servers that reference rules

Once you have created a rule, you must configure the virtual server to reference the rule. You can configure a virtual server to reference a rule by using either the Configuration utility or the **bigpipe** command.

To configure a virtual server that references a rule using the Configuration utility

1. In the navigation pane, click **Virtual Servers**.
   The Virtual Servers screen opens.
2. Add the attributes you want for the virtual server such as **address**, **port**, **unit ID**, and **interface**.
3. In the Resources section, click **Rule**.
4. In the list of rules, select the rule you want to apply to the virtual server.
5. Click the **Apply** button.

To configure a virtual server that references a rule from the command line

There are several elements required for defining a virtual server that references a rule from the command line:

```
b virtual <virt_serv_key> { <virt_options> <rule_name_reference> }
```

Each of these elements is described in Table 4.16.

Rule element	Description
<virt_serv_key>	A virtual server key definition: `<virtual_address>:<virt_port> [unit <ID>]`
<virt_options>	Virtual server options. For more information, see *Virtual server options*, on page 4-77.
<rule_name_reference>	A rule name reference. Rule names are strings of 1 to 31 characters. `use rule <rule_name>`

Table 4.16 *The command line rule elements*

Turning software acceleration off for virtual servers using IPFW rate filters

The software acceleration feature speeds packet flow for TCP connections when the packets are not fragmented. For configurations with no IPFW rate filter present, software acceleration is turned on by default by giving the global variable **fastflow_active** a default value of **auto**. The variable **fastflow_active auto** enables acceleration globally if IPFW filters are not present, and disables it globally if IPFW filters are present. (This is because, with acceleration on, IPFW only examines the first SYN packet in any given connection, rather than filtering all packets.) If you want to turn on acceleration globally but turn it off for the specific virtual servers that use IPFW rate filters, you must change the **fastflow_active** setting from **auto** to **on**, then disable the virtual servers individually using the **bigpipe virtual <ip>:<service> accelerate disable** command.

To set software acceleration controls from the command line

To enable software acceleration globally in a way that can be overridden for individual virtual servers, set the **bigpipe global** variable **fastflow_active** to **on** with the following command:

```
b global fastflow_active on
```

Then, to disable software acceleration for individual virtual servers that use IPFW rate filtering, use the following **bigpipe** command:

```
b virtual <ip>:<service> accelerate disable
```

For example, if you want to turn acceleration off for the virtual server **10.10.10.50:80**, type the following command:

```
b virtual 10.10.10.50:80 accelerate disable
```

You can define a virtual server with acceleration disabled using the following syntax:

```
b virtual <ip>:<service> use pool the_pool accelerate disable
```

For example, if you want to define the virtual server **10.10.10.50:80** with the pool **IPFW_pool** and acceleration turned off, type the following command:

```
b virtual 10.10.10.50:80 use pool IPFW_pool accelerate disable
```

Additional virtual server tasks

Once you have created a virtual server and configured options for it, you can perform the following tasks.

Enabling or disabling a virtual server

You can remove an existing virtual server from network service, or return the virtual server to service, using the **disable** and **enable** keywords. As an option, you can enable or disable a virtual server for a specific VLAN only.

When you disable a virtual server, the virtual server no longer accepts new connection requests, but it allows current connections to finish processing before the virtual server goes **down**.

To disable or enable a virtual server using the Configuration utility

1. In the navigation pane, click **Virtual Servers**.
   The list of virtual servers displays.
2. Click on the virtual server that you want to disable or enable.
   This displays the properties for that virtual server.
3. If you want to disable the virtual server for all VLANs, clear the **Enabled** check box. If you want to disable the virtual server for a specific VLAN, locate the **VLANs Disabled** box and move the relevant VLAN name from the **Existing** list to the **Disabled** list, using the arrows (>>).
4. If you want to enable the virtual server for all VLANs, click the **Enabled** check box (if not already checked). If you want to enable the virtual server for a specific VLAN, locate the **VLANs Disabled** box and move the relevant VLAN name from the **Disabled** list to the **Existing** list, using the arrows (>>).
5. Click Done.

◆ Note

*If the **Enabled** check box is checked and no VLANs are listed in the **Disabled** list of the **VLANs Disabled** box, the virtual server is enabled for all VLANs. If the **Enabled** check box is **not** checked, the virtual server is disabled for all VLANs.*

To disable or enable a virtual server from the command line

Use the following syntax to disable a virtual server from network service:

```
b virtual <virt_ip>:<service> [...<virt_ip>:<service>] disable
```

If you want to disable or enable a virtual server for one or more specific VLANs only, use the following syntax:

```
b virtual <virt_ip>:<service> vlans <vlan_list> disable | enable
```

Use the following syntax to return a virtual server to network service:

```
b virtual <virt_ip>:<service> enable
```

Note

If you do not specify a VLAN name with the b virtual command, the virtual server is enabled or disabled on all VLANs.

Enabling or disabling a virtual address

You can remove an existing virtual address from network service, or return the virtual address to service, using the **disable** and **enable** keywords. Note that when you enable or disable a virtual address, you inherently enable or disable all of the virtual servers that use the virtual address.

```
b virtual <virt_ip> disable
```

Use the following syntax to return a virtual address to network service:

```
b virtual <virt_ip> enable
```

Displaying information about virtual addresses

You can also display information about the virtual addresses that host individual virtual servers. Use the following syntax to display information about one or more virtual addresses included in the configuration:

```
b virtual <virt_ip> [... <virt_ip> ] show
```

The command displays information such as the virtual servers associated with each virtual address, the status, and the current, total, and maximum number of connections managed by the virtual address since the BIG-IP was last rebooted, or since the BIG-IP became the active unit (redundant configurations only).

Deleting a virtual server

Use the following syntax to permanently delete one or more virtual servers from the BIG-IP configuration:

```
b virtual <virt_ip>:<service> [... <virt_ip>:<service>] delete
```

Resetting statistics for a virtual server

Use the following command to reset the statistics for an individual virtual server:

```
b virtual [<virt_ip:port>] stats reset
```

Using other BIG-IP features with virtual servers

After you create a pool and define a virtual server that references the pool, you can set up additional features, such as network address translation (NATs) or extended content verification (ECV). For details on network address translations, see *NATs*, on page 4-132. For details on persistence for connections that should return to the node to which they last connected, see *Persistence*, on page 4-21.

Proxies

BIG-IP supports two types of proxies--An SSL Accelerator proxy, and a content converter proxy. Using either the Configuration utility or the **bigpipe proxy** command, you can create, delete, modify, or display the SSL or content converter proxy definitions on the BIG-IP.

For detailed information about setting up the SSL Accelerator feature, see the ***BIG-IP Solutions Guide,*** Chapter 9, *Configuring an SSL Accelerator*. For detailed information about setting up the content converter feature, see the ***BIG-IP Solutions Guide,*** Chapter 14, *Configuring a Content Converter*.

The SSL Accelerator proxy

The SSL Accelerator feature allows the BIG-IP to accept and terminate any connections that are sent via a fully-encapsulated SSL protocol. For example, the BIG-IP can accept HTTPS connections (HTTP over SSL), connect to a web server, retrieve the page, and then send the page to the client.

A key component of the SSL Accelerator feature is that the BIG-IP can retrieve the web page using an unencrypted HTTP request to the content server. With the SSL Accelerator feature, you can configure an SSL proxy on the BIG-IP that decrypts HTTP requests that are encrypted with SSL. Decrypting the request offloads SSL processing from the servers to the BIG-IP. This also allows the BIG-IP to use the header of the HTTP request to intelligently control how the request is handled. (You can optionally configure requests to the servers to be re-encrypted to maintain security on the server side of the BIG-IP as well, using a feature called ***SSL-to-server***. While the servers must then handle the final decryption and re-encryption, SSL processing is still faster than if the entire task were left to the servers.)

When the SSL proxy on the BIG-IP connects to the content server, and address translation is not enabled, the proxy uses the original client's IP address and port as its source address and port. In doing so, the proxy appears to be the client, for logging purposes.

BIG-IP offers several options for configuring an SSL Accelerator proxy. These options are configured separately for each SSL proxy that you create. You can configure these options at the time that you create the proxy.

◆ Note

Before configuring an SSL proxy, you must either obtain a valid x509 certificate from a Trusted certificate authority, or generate a valid temporary certificate. In either case, this certificate file must be in PEM format.

Table 4.17 lists the configurable SSL proxy options.

Options	Description
SSL-to-Server configuration	Causes the BIG-IP to re-encrypt decrypted requests before sending them to the server, as a way to maintain server-side security.
Client-side authentication	Allows you to configure the SSL proxy to either request, require, or ignore certificates presented by a client.
Server-side authentication	Allows you to configure certificate authentication between the SSL proxy and a content server. This capability is part of the SSL-to-Server feature.
HTTP header insertion	Allows you to configure an SSL proxy to insert various types of headers into HTTP requests. For more information, see Inserting headers into HTTP requests.
Specification of ciphers and protocol versions	Allows you to configure an SSL proxy to require specific ciphers or protocol versions. For more information, see Specifying SSL ciphers and protocol versions.
Configuration of trusted CAs	Allows you to configure certificate chaining and verification, as well as to configure the proxy to send to a client a list of CAs that the proxy trusts.
Rewriting of HTTP redirection	Allows you to configure the proxy to convert HTTP redirects to HTTPS redirects.
SSL session cache configuration	Allows you to configure the proxy to set a timeout value and a size for the SSL session cache.
SSL proxy failover configuration	Allows you to configure the proxy to initiate a failover on a redundant BIG-IP in the event of a fatal cryptographic hardware failure.
Shutdown configuration	Allows you to configure the way in which the proxy manages clean and unclean shutdowns of SSL connections.
Disabling of arp requests	Allows you to disable the proxy address for ARP requests.
lasthop pool configuration	Allows you to add a last hop pool to an SSL proxy.
proxy deletion	Allows you to delete an SSL proxy.

Table 4.17 *Configuration options for the SSL Accelerator*

Creating an SSL Accelerator Proxy

When creating an SSL Accelerator proxy, you can enable the proxy to handle either client-side SSL connections only, or both client-side and server-side SSL connections. The following procedures describe how to configure the SSL proxy for client-side connections only. To configure the proxy for server-side connections, see *Configuring SSL-to-Server*, on page 4-90.

To create an SSL proxy using the Configuration utility

1. In the navigation pane, click **Proxies**.
   The Proxies screen opens.
2. Click the **ADD** button.
   The Add Proxy screen opens.
3. In the Proxy Type field, check the box labeled **SSL**.
4. Configure the remaining attributes that you want to use with the proxy
5. Click **Done**.

To create an SSL proxy from the command line

Use the following command-line syntax to create an SSL proxy:

```
b proxy <ip>:<service> [unit <unit_id>] \
target <server|virtual> <ip>:<service> \
clientssl enable \
clientssl key <clientssl_key> \
clientssl cert <clientssl_cert>
```

The following example creates an SSL proxy:

```
b proxy 10.1.1.1:443 unit 1 \
target virtual 20.1.1.1:80 \
clientssl enable \
clientssl key my.server.net.key \
clientssl cert my.server.net.crt
```

When the SSL proxy is written in the **/config/bigip.conf** file, it looks like the sample in Figure 4.42.

```
proxy 10.1.1.1:443 unit 1 {
    target virtual 20.1.1.1:http
    clientssl enable
    clientssl key my.server.net.key
    clientssl cert my.server.net.crt
}
```

Figure 4.42 *An example of an SSL proxy configuration*

Configuring SSL-to-Server

Once the SSL Accelerator proxy has decrypted a client request, you might want the BIG-IP to re-encrypt that request before it sends the request to the server, to maintain server-side security. This feature is known as ***SSL-to-Server***. To implement this feature, you can use either the Configuration utility or the command line.

◆ Note

The SSL-to-server feature requires that you create an SSL proxy, described in Creating an SSL Accelerator Proxy, on page 4-88.

Enabling the SSL-to-Server option

You can configure SSL-to-Server using either the Configuration utility or the command line.

To configure SSL-to-Server using the Configuration utility

1. In the navigation pane, click **Proxies**.
   The Proxies screen opens.
2. Click the **ADD** button.
   The Add Proxy screen opens.
3. In the Proxy Type box, check the boxes labeled **SSL** and **ServerSSL**.
4. Configure the remaining attributes that you want to use with the SSL proxy and the SSL-toServer feature.

To configure SSL-to-Server from the command line

Use a command such as in the following example to create an SSL-to-server proxy:

```
b proxy 10.1.1.1:443 \
target virtual 20.1.1.10:443 \
clientssl enable \
clientssl key my.server.net.key \
clientssl cert my.server.net.crt \
serverssl enable
```

You must either configure trusted server-side Certificate Authorities or configure the SSL proxy to ignore server-side certificates. For more information, see *Configuring server certificate authentication*, on page 4-92.

Figure 4.43 shows the state of the **/config/bigip.conf** file, after creating an SSL proxy with SSL-to-Server enabled. Note that the certificate and key files for client-side SSL connections have also been configured.

```
proxy 10.1.1.1:443 unit 1 {
    target virtual 20.1.1.1:https
    clientssl enable
    clientssl key my.server.net.key
    clientssl cert my.server.net.crt
    serverssl enable
}
```

Figure 4.43 *SSL proxy entries in /**config/bigip.conf** wth SSL-to-Server enabled*

Configuring client certificates

This option extends the SSL-to-Server feature by allowing the BIG-IP to authenticate itself using client certificates. You can thus specify a key file and a certificate file for the proxy as an SSL client, as it acts on the server side. When a server-side SSL certificate is specified, the certificate is used only if the server requests client authentication.

To configure client certificates using the Configuration utility

Configuring client certificates for SSL-to-Server using the Configuration utility is similar to configuring the existing client SSL key/certificate pair.

1. From navigation pane, click **Proxies**.
2. Click the **Add** button.
3. Check the **SSL** and **ServerSSL** check boxes.
4. In the boxes labeled **Server SSL Certificate** and **Server SSL Key,** either type the names of the key and certificate files or select the names from a list of available key and certificate files.
5. Click **Done**.

To configure client keys and certificates from the command line

When configuring SSL-to-Server, you can use the **bigpipe proxy** command to designate a key file and a certificate file. This is done as is the following example:

```
b proxy 10.1.1.1:443 \
target virtual 20.1.1.10:443 \
clientssl enable \
clientssl key my.server.net.key \
clientssl cert my.server.net.crt \
```

```
serverssl enable \
serverssl key my.client.net.key \
serverssl cert my.client.net.crt
```

Figure 4.44 shows the state of the **/config/bigip.conf** file, after both creating an SSL proxy with SSL-to-Server enabled and configuring the certificates and keys for both client-side and server-side SSL connections.

```
proxy 10.1.1.1:443 unit 1 {
    target virtual 20.1.1.1:https
    clientssl enable
    clientssl key my.server.net.key
    clientssl cert my.server.net.crt
    serverssl enable
    serverssl key my.client.net.key
    serverssl cert my.client.net.crt
}
```

Figure 4.44 *SSL proxy entries in* ***/config/bigip.conf*** *with server-side certificate and key files configured*

Configuring server certificate authentication

You can verify server certificates, as well as specify the maximum number of certificates to be traversed in a server certificate chain.

◆ Tip

In addition to configuring certificate authentication, you must also configure the trusted CAs. For more information, see Specifying a list of trusted Certificate Authorities (CAs), on page 4-102.

Verifying server certificates

To implement certificate authentication on the server side (that is, between the SSL proxy and the server), you can configure the proxy to either require the server to present a certificate or ignore the presentation of a certificate. Note, however, that you cannot require the server to present a certificate if anonymous cipher suites are negotiated.

If this option is set to **require** (the default setting), the proxy verifies any certificate presented by the server. If this verification fails, the SSL connection also fails, and the corresponding client connection is closed. If this option is set to **ignore**, verification fails only when a certificate is presented by the server and the certificate is expired or malformed.

To verify server certificates using the Configuration utility

1. From navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Server Certificate** field, select **require** or **ignore** from the box.
4. Click **Done**.

To verify server certificates from the command line

This option is specified as **serverssl server cert** on the **bigpipe proxy** command line. The following command shows an example.

```
b proxy <ip>:<service> serverssl server cert require
```

Specifying traversal of certificate chains

In addition to the option to require or ignore a certificate presented by the server, SSL-to-Server has an option to specify the maximum number of certificates that can be traversed in a server certificate chain.

To configure certificate traversal using the Configuration utility

1. From navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Authentication depth** box, type a whole number. The default setting is **9**.
4. Click **Done**.

To configure certificate traversal from the command line

On the **bigpipe proxy** command line, this option is specified as **serverssl authenticate depth**, followed by a whole number representing the maximum number of certificates to be traversed. The following command shows an example.

```
b proxy <ip>:<service> serverssl authenticate depth 8
```

Configuring client-side authentication

This feature offers several options pertaining to client authentication. First, you can set the basic authentication option, which determines the extent to which an SSL proxy authenticates a client. Second, you can configure the SSL proxy to authenticate a client either once per SSL session or also upon each subsequent reuse of the session. Finally, you can specify the maximum number of certificates to be traversed in a client certificate chain. The following two sections explain these options.

◆ Tip

In addition to configuring certificate authentication, you must also configure the trusted CAs. In so doing, it is recommended that you also configure the list of advertised CAs, to ensure that all clients know which CAs are trusted by the proxy. *For more information, see Advertising a Trusted CA list, on page 4-104.*

Basic authentication options

You can configure an SSL proxy to handle authentication of clients in three ways:

- You can configure the proxy to **request** and verify a client certificate. In this case, the SSL proxy always grants access regardless of the status or absence of the certificate.
- You can configure the proxy to **require** a client to present a valid and trusted certificate before granting access.
- You can configure the proxy to **ignore** a certificate (or lack of one) and therefore never authenticate the client. This is the default setting.

◆ Tip

*The **request** option works well with the header insertion feature. Configuring the SSL proxy to insert client certificate information into an HTTP client request and to authenticate clients based on the **request** option allows the BIG-IP or a server to then perform actions such as redirecting the request to another server, or sending different content back to the client.*

To configure client-side authentication using the Configuration utility

1. From the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Client Certificate** box, choose either the **Request**, **Require**, or **Ignore** option.
4. Click **Done**.

To configure client-side authentication from the command-line

To configure client-side authentication from the command line, use the **bigpipe proxy** command and specify the desired option, as follows:

```
b proxy <ip>:<service> [clientssl] client cert <request | require | ignore>
```

Additional authentication options

If an SSL proxy is configured to verify client certificates, you can use two other options to configure client authentication in more detail: per-session authentication and authentication depth.

Per-session authentication

You can configure an SSL proxy to require authentication either once per SSL session, or once per session and upon each subsequent reuse of an SSL session. The default setting for this option is **once**, which causes the SSL proxy to request a client certificate and authenticate the client once per session.

To modify per-session authentication using the Configuration utility

You can modify the SSL proxy to require authentication not only once per session, but also upon each subsequent reuse of an SSL session.

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. Click on the **Client Authenticate Once** box. This changes the setting from **once** to **always**.
4. Click **Done**.

To modify per-session authentication from the command line

To modify the SSL proxy to require authentication not only once per session, but also upon each subsequent reuse of an SSL session, specify **always** argument with the **bigpipe proxy** command, as follows. This changes the setting from **once** to **always**.

```
bigpipe proxy <ip>:<service> [clientssl] authenticate <once | always>
```

Authentication depth

Using this option, you can configure the maximum number of certificates that can be traversed in the client certificate chain. The default value is nine. If a longer chain is provided, and the client has not been authenticated within this number of traversals, client certificate verification fails.

To configure authentication depth using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Client Authenticate Depth** box, type a whole number.
4. Click **Done**.

To configure authentication depth from the command line

To configure authentication depth from the command line, use the **authenticate depth** argument with the **bigpipe proxy** command, as follows:

```
b proxy <ip>:<service> [clientssl] authenticate depth <num>
```

Inserting headers into HTTP requests

You can configure the SSL proxy to insert several kinds of headers into an HTTP client request. They are:

- A custom HTTP header
- Cipher specification

- Client certificate fields
- Client session IDs

An example of when you might want to insert a header into an HTTP request is when the proxy is configured to request, rather than require, a certificate during client authentication. Because client authentication always succeeds in this case, despite status of the certificate, you might want the proxy to insert information into the HTTP request about the client certificate and the results of the verification attempt. Based on this information, the BIG-IP or a server could then perform actions such as redirecting the request to another server, or sending different content back to the client.

If any of these header types is inserted into a valid HTTP request, the SSL proxy places the header so that it immediately follows the first line of the request. If more than one header is inserted, the headers are inserted in the order listed above.

The following sections describe these header types.

Sometimes, the SSL proxy might not be able to insert a specified header, such as when a client request uses an non-standard HTTP method, or the request is pipelined. If you are making security decisions based on the value of these headers, you should exercise caution. Also, using the HTTP header insertion feature is not recommended when the SSL proxy targets a virtual server directly accessible to client connections. This is because it is not known whether the connection has come through the SSL proxy or directly from the client.

A custom HTTP header

When adding an SSL proxy, you can configure the proxy to insert a string of your choice into an HTTP request. This feature is useful for custom applications, as well as for securing content. For example, when the Outlook Web Access application detects the presence of a particular custom HTTP header, the application generates embedded URLs using the protocol HTTPS instead of HTTP.

A properly-formatted custom HTTP header is in the form of **Field: Value**. Note that an improperly-formatted custom HTTP header could cause the content server to fail in its handling of proxied requests.

To insert a custom header using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Insert HTTP Header String** box, type a custom HTTP header in the form of **Field: Value**.
4. Click **Done**.

To insert a custom header from the command line

To insert a custom header into an HTTP request using the command line, specify the **header insert** argument with the **bigpipe proxy** command, as follows:

```
b proxy <ip>:<service> header insert \"quoted string\"
```

A cipher specification

When adding an SSL proxy, you can configure the proxy to insert information about the negotiated SSL cipher into an HTTP request. When you configure this option, the SSL proxy inserts the actual cipher name, the SSL version, and the number of significant bits into the HTTP request.

A properly-formatted cipher specification header is in the form **SSLClientCipher: [cipher] version=[version] bits=[bits]**, where **[cipher]**, **[version]**, and **[bits]** represent the actual cipher name, version, and number of significant cipher bits, respectively.

The ability to insert a cipher specification into a client request is useful for two primary reasons:

- Inserting cipher information into an HTTP request can ensure that a client uses a specific cipher strength, thus enhancing the security of the SSL connection. Also, if the cipher strength of the client is unacceptable, you can direct them to a "cipher upgrade" path, rather than discarding the session altogether.
- You can create rules that perform load balancing based on the cipher strength specified in the inserted header. Thus, using the HTTP request string variable **http_header**, you could create a rule such as that shown in Figure 4.45.

```
if (exists http_header "SSLClientCipher") {
    if (http_header "SSLClientCipher" contains "bits=128") {
       use ( secure_pool )
    }
    else {
       redirect to "<https://%h/upgradebrowser.html>"
    }
}
else {
    redirect to "<https://%h/servererror.html>"
}
```

Figure 4.45 *A rule based on cipher strength specified in an HTTP header*

To insert a cipher specification using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. Check the **Insert Cipher** check box.
4. Click **Done**.

To insert a cipher specification from the command line

Specify the **cipher insert** argument with the **bigpipe proxy** command, as follows:

```
b proxy <ip>:<service> [clientssl] cipher insert <enable | disable>
```

Client certificate fields

When adding an SSL proxy, you can configure the proxy to insert into an HTTP request a header for each field of a client certificate. This feature is most useful when:

- You have configured the SSL proxy to authenticate clients with the **request** option. For more information, see *Configuring client-side authentication*, on page 4-93.
- You want to better control the load balancing of your network traffic. In this case, you can create a rule that performs load balancing according to the certificate information in the header. Figure 4.46 shows an example.

```
if (exists http_header "SSLClientCertStatus") {
    if (http_header "SSLClientCertStatus" contains "OK") {
       use ( authenticated_pool )
    }
    else {
       redirect to "<https://%h/authenticationfailed.html>"
    }
}
else {
    redirect to "<https://%h/servererror.html>"
}
```

Figure 4.46 *A rule based on certificate status specified in an HTTP header*

Table 4.18 shows the client certificate headers that the SSL proxy can insert into a client request. For each header, the required format, description, and keyword is shown.

Header Name	Required Format	Description
Certificate status	SSLClientCertStatus: [status]	The status of the client certificate. The value of [status] can be "**NoClientCert**", "**OK"**, or "**Error"**. If status is "**NoClientCert"**, only this header is inserted into the request. If status is "**Error**", the error is followed by a numeric error code.
Certificate version	SSLClientCertVersion: [version]	The version of the certificate.
Certificate serial number	SSLClientCertSerialNumber: [serial]	The serial number of the certificate.
Signature algorithm of the certificate	SSLClientCert: [alg]	The signature algorithm of the certificate.
Issuer of the certificate	SSLClientCert: [issuer]	The issuer of the certificate.

Table 4.18 *Required formats of client certificate headers*

Header Name	Required Format	Description
Certificate validity dates	SSLClientCert: [before] SSLClientCert: [after]	The validity dates for the certificate. The certificate is not valid before or after the dates represented by [before] and [after], respectively.
Certificate subject	SSLClientCert: [subject]	The subject of the certificate.
Public key of the subject	SSLClientCert: [key]	The type of public key type. The allowed types are "**RSA ([size] bit)"**, "**DSA"**, or "**Unkown public key**".
The certificate itself	SSLClientCert: [cert]	The actual client certificate.
MD5 hash of the certificate	SSLClientCert: [hash]	The MD5 hash of the client certificate.

Table 4.18 *Required formats of client certificate headers*

To insert fields of a client certificate using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Insert Certificate** box, check the appropriate check boxes.
4. Click **Done**.

To insert fields of a client certificate from the command line

To insert headers for the fields of a client certificate into an HTTP request using the command line, specify the **client cert insert** argument with the **bigpipe proxy** command, as follows:

```
b proxy <ip>:<service> [clientssl] client cert insert
<([versionnum] [serial] [sigalg] [issuer] [validity] [subject]
[subpubkey] [whole] [hash])+ | disable>
```

Client session IDs

When adding an SSL proxy, you can configure the proxy to insert a client SSL session ID header into an HTTP request.

The header that is inserted can be one of two types:

- A header in which the session ID is the session ID initially negotiated with the client for the corresponding TCP connection. The proper format of this header is **SSLClientSessionID:X**, where **X** represents the hexidecimal representation of the SSL session ID that was initially negotiated with the client for the corresponding TCP connection.

- A header in which the session ID is the current session ID. The proper format of this header is **SSLClientCurrentSessionID:X**, where **X** represents the current SSL session ID.

If you enable the insertion of session ID headers, but specify neither of these two types of session IDs, the SSL proxy inserts the session ID initially negotiated with the client.

To insert a session ID header using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Insert Client Session ID** box, check either or both of the **Initial** and **Current** check boxes.
4. Click **Done**.

To insert a session ID header from the command line

To insert a session ID header into an HTTP request using the command line, specify the **sessionid insert** argument with the **bigpipe proxy** command, as follows:

```
b proxy <ip>:<service> [clientssl] sessionid insert [initial] [current] [enable]
```

◆ Note

One use of client session IDs is to enable SSL persistence. Note that SSL persistence should not be enabled on pools that load balance plain-text traffic, that is, traffic resulting from SSL proxies on which SSL termination is enabled.

Specifying SSL ciphers and protocol versions

For each SSL proxy, you can specify both the ciphers available for SSL connections, and the protocol versions that are not allowed.

When configuring ciphers and protocol versions, you must ensure that the ciphers and the protocol versions configured for the SSL proxy match those of the proxy's peer. That is, ciphers and protocol versions for the client-side SSL proxy must match those of the client, and ciphers and protocol versions for the server-side SSL proxy must match those of the server.

For example, a client might connect to and successfully establish an SSL connection to an SSL proxy that is configured to use both client-side and server-side SSL. After the client sends additional data (such as an HTTP request), the SSL proxy attempts to establish an SSL connection to a server. However, the SSL proxy might be configured to enable only 3DES ciphers for server-side SSL, and the servers might be configured to accept only RC4 ciphers. In this case, the SSL handshake between the SSL proxy and the server will fail because there are no common ciphers enabled. This results in

the client connection being closed. If the client is using a browser, the user will likely receive an error message indicating that the web page failed to load.

The following sections describe how to configure cipher lists and protocol versions for the SSL proxy.

Configuring cipher lists

You can configure the list of SSL ciphers that are available for both client-side and server-side SSL connections. Whether using the Configuration utility or the **bigpipe proxy** command, you can specify a string to indicate the available list of SSL ciphers.

◆ Note

To see the complete syntax for the cipher list, see the following OpenSSL web site: ***http://www.openssl.org/docs/apps/ciphers.html****.*

To configure a cipher list using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In either or both of the **Client Cipher List String** or **Server Cipher List String** boxes, type a properly-formatted string.
4. Click **Done**.

To configure a cipher list from the command line

To specify a list of ciphers from the command line, specify the client-side cipher list or the server-side cipher list, as follows:

```
b proxy <ip>:<service> [clientssl] ciphers \"quoted string\"
b proxy <ip>:<service> serverssl ciphers \"quoted string\"
```

◆ Tip

You can use the ***openssl ciphers*** *command to test the validity of a cipher string.*

Configuring invalid protocol versions

For both client-side and server-side SSL connections, you can specify SSL protocol versions that should not be allowed. You can declare up to two of the following three protocol versions to be invalid: **SSLv2**, **SSLv3**, and **TLSv1**. If no protocol versions are specified, all SSL protocol versions are allowed. If all three protocol versions are disallowed, no SSL sessions can be successfully negotiated.

To specify invalid protocol versions using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Client-side Connections Do Not Use These SSL Versions** box or the **Server-side Connections Do Not Use These SSL Versions** box, check the appropriate check boxes.
4. Click **Done**.

To specify invalid SSL protocol versions from the command line

Use the following syntax:

```
b proxy <ip>:<service> [clientssl] invalid [SSLv2] [SSLv3] [TLSv1]
b proxy <ip>:<service> serverssl invalid [SSLv2] [SSLv3] [TLSv1]
```

Specifying a list of trusted Certificate Authorities (CAs)

For both client-side and server-side SSL connections, you can specify trusted certificate authorities (CAs). The proxy can then use this CA specification to do the following:

- Build certificate chains
- Verify client certificates
- Advertise to clients the CAs that the server trusts

The following sections describe each of these uses of the trusted CAs list.

Building a certificate chain

Sometimes, a certificate that the SSL proxy uses to authenticate itself to a peer is signed by an intermediate CA that is not trusted by that peer. In this case, the proxy might need to build a certificate chain. The proxy allows you to build a certificate chain by specifying the name of a specific certificate chain file, either through the Configuration utility or from the command line. Note that the certificate files that make up the chain file must be in PEM format.

When attempting to access the specified chain file, the SSL proxy searches for the file in the following manner:

1. The proxy looks to see that the specified file has a **.chain** extension.
2. If the file specification does not include a **.chain** extension, the proxy appends that extension to the file and then searches for the file.
3. If the file is not found, the proxy instead appends a **.crt** extension to the file and searches again.

4. If the file is still not found, the proxy uses the same file name as that of the configured certificate. For example, the proxy might take the file name **www.dot.com.crt**, replace the **.crt** file name extension with the **.chain** extension, and search on the file name **www.dot.com.chain**.
5. If unable to build the certificate chain using the preceding procedure, the proxy attempts to build the chain through certificate verification, described in the following section.

To build a certificate chain using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the box **Client Chain File** or **Server Chain File**, either select the name of a Trusted CAs file from the box, or type the name of a Trusted CA file.
4. Click **Done**.

To build a certificate chain from the command line

To build a certificate chain from the command line, type the **bigpipe proxy** command with the appropriate arguments, as follows:

```
b proxy <ip>:<service> [clientssl] chain <clientside chain file name>
b proxy <ip>:<service> serverssl chain <serverside chain file name>
```

Verifying certificates

For both client-side and server-side SSL processing, you can configure the SSL proxy to verify certificates. Using either the Configuration utility or the **bigpipe proxy** command, you can specify both a Trusted CA file name and a Trusted CA path name, which the proxy then uses to verify client certificates.

Certificate verification is useful for the following reasons:

- To authenticate the proxy's peer
- To build a certificate chain to be sent to a peer, when the standard method for building a certificate chain fails

The Trusted CA file.
The Trusted CA file that you specify to configure certificate verification contains one or more certificates, in PEM format. If you do not specify a Trusted CA file, or the specified Trusted CA file is not accessible to the proxy, the proxy uses the default file name **/config/bigconfig/ssl.crt/intermediate-ca.crt**.

The Trusted CA path.
When searching a Trusted CA path, the proxy only examines those certificates that include a symbolic link to a certificate file. To ensure that each certificate has a link to its corresponding certificate file, you can

configure the proxy to generate these symbolic links. If you do not specify a Trusted CA path, or the Trusted CA path is not accessible to the proxy, the proxy uses the default path name **/config/bigconfig/ssl.crt/**.

Note that each certificate file should contain only one certificate. This is because only the first certificate in the file is used.

To specify the Trusted CA file and Trusted CA path using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the boxes **Client Trusted CA File** and **Client Trusted CA Path**, or **Server Trusted CA File** and **Server Trusted CA Path,** either select the name of a Trusted CAs file and path from the box, or type the name of a Trusted CA file or path.
4. If you want to ensure that each certificate has a link to its corresponding file, check the **Generate Symbolic Links for Client Trusted CAs Path** check box.
5. Click **Done**.

To specify the Trusted CA file and Trusted CA path from the command line

To specify the Trusted CA file and Trusted CA path from the command line, type the **bigpipe proxy** command, using the appropriate arguments, as follows:

```
b proxy <ip>:<service> [clientssl] ca file <clientside CA file name>
b proxy <ip>:<service> [clientssl] ca path <clientside CA path name>
b proxy <ip>:<service> serverssl ca file <serverside CA file name>
b proxy <ip>:<service> serverssl ca path <serverside CA path name>
```

Advertising a Trusted CA list

If you intend to configure the SSL proxy to require or request client certificates for authentication, you usually want the proxy to send to clients a list of CAs that the server is likely to trust. Although modern browsers automatically limit the user's selection of trusted CAs based on the proxy's configured list of trusted CAs, older browser versions may not have this capability.

The list of advertised trusted CAs can be different from the actual Trusted CA file configured as part of certificate verification.

To configure the proxy to send this list, you can specify a PEM-formatted certificate file that contains one or more CAs that a server trusts for client authentication. If no certificate file is specified, no list of trusted CAs is sent to a client.

To advertise a list of trusted CAs using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Client Certificate CA File** box, select a file name from the box, or type the certificate CA file name.
4. Click **Done**.

To advertise a list of trusted CAs from the command line

To configure the proxy to send a list of trusted CAs to a client from the command line, type the **bigpipe proxy** command, using the following arguments:

```
b proxy <ip>:<service> [clientssl] client cert ca <clientside client cert CA file name>
```

Rewriting HTTP redirection

When a client request is redirected from the HTTPS to the HTTP protocol, an SSL proxy can rewrite that redirection to HTTPS. (Specifically, this applies to HTTP responses 301, 302, 303, 305, and 307). This ability for the SSL proxy to rewrite HTTP redirections provides additional security by ensuring that client requests remain on a secure channel.

Another benefit of the ability to rewrite HTTP redirection pertains to IIS and Netscape web-server environments. Prior to this feature, a web server running IIS and Netscape would redirect a request incorrectly if the original request included a malformed directory name (without a trailing slash [/]). The ability for an SSL proxy to rewrite such a redirection solves this problem.

◆ Note

If your web server is an IIS server, you can configure that server, instead of the SSL proxy, to handle any rewriting of HTTP redirections. To solve the problem described above, you can install a special BIG-IP file, ***redirectfilter.dll****, on your IIS server. For more information, see Rewriting HTTP redirection, on page 4-41.*

Note that the rewriting of any redirection only takes place in the HTTP **Location** header of the redirection response, and not in any content of the redirection.

This rewrite feature can rewrite the protocol name and the port number. Optionally, you can specify how the proxy should handle URIs during a rewrite.

Rewriting the protocol name

This feature allows the SSL proxy to rewrite the HTTP protocol name to HTTPS. For example, a client might send a request to **https://www.sample.com/bar** and be initially redirected to **http://www.sample.com/bar/**, which is a non-secure channel. If you want the client request to remain on a secure channel, you can configure the SSL proxy to rewrite the redirected URI to go to **https://www.sample.com/bar/** instead. (Note the addition of the trailing slash.)

Rewriting the port number

In addition to being able to rewrite the protocol name from HTTP to HTTPS, the SSL proxy can also rewrite the port number of the redirected request. This happens in the case when the web server and/or SSL proxy are listening on a non-standard port, for example, when the client request is initially redirected to **http://www.sample.com:8080/bar/**. In this case, the SSL proxy rewrites not only the protocol name but the port number also. If, however, the SSL proxy is listening on the standard HTTPS port **443**, then the SSL proxy removes the **8080** port number, without replacing it with **443**.

Selecting URIs to rewrite

When configuring the SSL proxy to rewrite HTTP redirections, you can specify whether the proxy should rewrite only those URIs **matching** the URI originally requested by the client (minus the trailing slash), or **all** URIs. In the latter case, the SSL proxy always rewrites redirected-to URIs, and rewrites those URIs as if they matched the originally-requested URIs.

Table 4.19 shows examples of how redirections of client requests are transformed when the SSL proxy is listening on port **443** and the rewrite feature is enabled.

Original redirection	Rewrite of Redirection with SSL Proxy Listening on Port 443
http://www.myweb.com/myapp/	https://www.myweb.com/myapp/
http://www.myweb.com:8080/myapp/	https://www.myweb.com/myapp/

Table 4.19 *Examples of rewriting HTTP redirections with SSL proxy listening on port* ***443***

Table 4.20 shows examples of how redirections of client requests are transformed when the SSL proxy is listening on port **4443** and the rewrite feature is enabled.

Original redirection	Rewrite of Redirection with SSL Proxy Listening on Port 4443
http://www.myweb.com/myapp/	https://www.myweb.com:4443/myapp/
http://www.myweb.com:8080/myapp/	https://www.myweb.com:4443/myapp/

Table 4.20 *Examples of rewriting HTTP redirections with SSL proxy listening on port* ***4443***

To configure the rewrite feature using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Rewrite Redirects** box, if you want to enable the feature, select either **Matching** or **All**. from the list. To disable the feature, do not select an option from the box. By default, the feature is disabled.
4. Click **Done**.

To configure the rewrite feature from the command line

To configure this feature from the command line, type the **bigpipe proxy** command and specify the **redirects rewrite** argument, as follows:

```
b proxy <ip>:<service> redirects rewrite <<matching | all> [enable] | disable>
```

Configuring SSL session cache

For both client-side and server-side SSL connections, you can configure timeout and size values for the SSL session cache.

Because each proxy maintains a separate client-side SSL session cache, the client-side values can be configured on a per-proxy basis. For server-side SSL connections, however, the proxy maintains a single session cache. Thus, server-side session cache values must be configured globally.

Setting SSL Session Cache Timeout

Using either the Configuration utility or the **bigpipe** command, you can specify the number of usable lifetime seconds of negotiated SSL session IDs. The default timeout value for the SSL session cache is 300 seconds. Acceptable values are integers greater than or equal to 5.

Clients attempting to resume an SSL session with an expired session ID are forced to negotiate a new session.

Client-side timeout values. The client-side timeout values are configured on a per-proxy basis. Client-side timeout values can be set to zero, which represents no timeout.

If the timeout value for the client-side SSL session cache is set to zero, the SSL session IDs negotiated with that proxy's clients remain in the session cache until either the proxy is restarted, or the cache is filled and the purging of entries begins. Setting a value of zero can introduce a significant security risk if valuable resources are available to a client that is reusing those session IDs. It is therefore common practice to set the SSL session cache timeout to a length of time no greater than 24 hours and for even shorter periods.

Server-side timeout values. A single, server-side timeout value is configured globally. This timeout value cannot be set to zero. For optimal performance, the timeout value should be set to the minimum SSL session cache timeout value used by the servers to which the proxy makes server-side SSL connections. Under certain conditions, the proxy attempts to efficiently negotiate a new server-side SSL session prior to its expiration.

To set the timeout value of the client-side SSL session cache using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Client Session Cache Timeout** box, type an integer greater than or equal to 5, or use the default value.
4. Click **Done**.

To set the timeout value of the server-side SSL session cache using the Configuration utility

1. In the navigation pane, click **System**.
2. Click on the Advanced Properties tab.
3. In the **Server SSL Session Cache Timeout** box, type zero or an integer greater than or equal to five, or use the default value.
4. Click **Done**.

To set the timeout value of the client-side SSL session cache from the command line

To set the timeout value of the client-side SSL session cache, type the **bigpipe proxy** command with the appropriate arguments, as follows:

```
b proxy <ip>:<service> [clientssl] cache timeout <num>
```

To set the timeout value of the server-side SSL session cache from the command line

To set the timeout value of the server-side SSL session cache, type the **bigpipe global** command with the appropriate arguments. as follows:

```
b global sslproxy serverssl cache timeout <num>
```

Setting SSL session cache size

Using either the Configuration utility or the **bigpipe** command, you can specify the maximum size of the SSL session cache. The default value for the size of the SSL session cache is 20,000 entries.

The client-side values for the maximum size of the session cache are configured on a per-proxy basis. A single, server-side value for the maximum size of the session cache is configured globally.

To set the maximum size of the client-side SSL session cache using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Client Session Cache Size** box, type an integer or use the default value.
4. Click **Done**.

To set the maximum size of the client-side SSL session cache size from the command line

To set the maximum size of the client-side SSL session cache from the command line, type the **bigpipe proxy** command with the appropriate arguments, as follows:

```
b proxy <ip>:<service> [clientssl] cache size <num>
```

To set the maximum size of the server-side SSL session cache using the Configuration utility

1. In the navigation pane, click **System**.
2. Click the Advanced Properties tab.
3. In the **Server SSL Session Cache Size** box, type an integer or use the default value.
4. Click **Done**.

To set the maximum size of the server-side SSL session cache from the command line

To set the maximum size of the server-side SSL session cache from the command line, type the **bigpipe global** command with the appropriate arguments. as follows:

```
b global sslproxy serverssl cache size <num>
```

Configuring SSL proxy failover

If you have a redundant BIG-IP configuration, you can configure the SSL proxy to initiate an automatic failover in the event of a fatal cryptographic hardware module failure. A ***fatal*** failure is the condition where the BIG-IP, after having had an initial success communicating with the cryptographic accelerator module, subsequently receives a hardware error.

This option is configured globally, and by default is set to **disable**.

Note

In redundant configurations, connections handled by the SSL proxy are not mirrored, and therefore cannot be resumed by the peer unit upon failover.

To configure SSL proxy failover using the Configuration utility

1. In the navigation pane, click **System**.
2. Click the Advanced Properties tab.
3. In the **Failover on SSL Accelerator Failure** box, check the **Enable** or **Disable** check box.
4. Click **Done**.

To configure SSL proxy failover from the command line

To enable or disable the SSL proxy for failover from the command line, type the **bigpipe global** command with the appropriate arguments, as follows:

```
b global sslproxy failover <enable | disable>
```

Configuring SSL shutdowns

With respect to the shutdown of SSL connections, you can configure two global options on the BIG-IP:

- Forcing clean SSL shutdowns
- Allowing SSL sessions to resume after unclean shutdown

The following sections describe these options.

Forcing clean SSL shutdowns

By default, the SSL proxy performs unclean shutdowns of all SSL connections, which means that underlying TCP connections are closed without exchanging the required SSL shutdown alerts. If you want to force the SSL proxy to perform a clean shutdown of all SSL connections, you can disable the default setting.

This feature is especially useful with respect to the Internet Explorer browser. Different versions of the browser, and even different builds within the same version of the browser, handle shutdown alerts differently. Some versions or builds require shutdown alerts from the server, while others do not, and the SSL proxy cannot always detect this requirement or lack of it. In the case where the browser expects a shutdown alert but the SSL proxy has not exchanged one (the default setting), the browser displays an error message.

To configure SSL shutdowns using the Configuration utility

1. In the navigation pane, click **System**.
2. Click the Advanced Properties tab.
3. In the **Force Unclean Shutdown Of All SSL Connections** box, check or clear the check box.
4. Click **Done**.

To configure SSL shutdowns from the command line

To configure SSL shutdowns from the command line, type the **bigpipe global** command with the appropriate arguments, as follows:

```
b global sslproxy unclean shutdown <enable | disable>
```

Resuming SSL sessions

In addition to forcing clean shutdowns, you can also configure the SSL proxy to prevent an SSL session from being resumed after an unclean shutdown. The default option is **disable**, which causes the SSL proxy to allow uncleanly shut down SSL sessions to be resumed. Conversely, when the **enable** option is set, the SSL proxy refuses to resume SSL sessions after an unclean shutdown.

To configure the SSL proxy to resume SSL sessions using the Configuration utility

You can allow the SSL proxy to resume, or prevent the SSL proxy from resuming, SSL sessions after an unclean shutdown.

1. In the navigation pane, click **System**.
2. Click the Advanced Properties tab.
3. In the **Do Not Resume Uncleanly Shutdown SSL Connections** box, check or clear the check box.
4. Click **Done**.

To configure the SSL proxy to resume SSL sessions from the command line

To allow the SSL proxy to, or prevent the SSL proxy from, resuming SSL sessions after an unclean shutdown from the command line, type the **bigpipe global** command with the appropriate arguments, as follows:

```
b global sslproxy strict resume <enable | disable>
```

Disabling ARP requests

By default, the BIG-IP responds to ARP requests for proxy address and sends a gratuitous ARP request for router table update. If you want to disable the proxy address for ARP requests, you must specify **arp disable**.

Adding a last hop pool to an SSL proxy

In cases where you have more than one router sending connections to a BIG-IP, connections are automatically sent back through the same router from which they were received when the **auto_lasthop** global variable is enabled, as it is by default. If the global **auto_lasthop** is disabled for any reason (for example, you may not want it for a virtual server), or if you want to exclude one or more routers from **auto_lasthop** you can direct your replies to the last hop router using a last hop pool. The lasthop pool will take precedence over **auto_lasthop**.

To configure a last hop pool, you must first create a pool containing the router inside addresses. After you create the pool, use the following syntax to configure a last hop pool for a proxy:

```
b proxy <ip>:<service> lasthop pool <pool_name>
```

For example, if you want to assign the last hop pool named **ssllasthop_pool** to the SSL proxy **11.12.1.200:443**, type the following command:

```
b proxy 11.12.1.200:443 lasthop pool ssllasthop_pool
```

Deleting an SSL proxy

If you want to delete the SSL proxy **209.100.19.22:80**, type a command such as the following:

```
b proxy 209.100.19.22:443 delete
```

The content converter proxy

The content converter proxy performs conversion of URLs to ***ARLs*** (Akamai Resource Locators). ARLs point to copies of URL targets that are stored on geographically nearby servers on the Akamai Freeflow Network™ for greater speed of access. The conversion from URL to ARL is performed whenever a client accesses a web page on a customer site containing a URL with an ARL counterpart, giving it the name *on-the-fly content conversion*. On-the-fly content conversion has the advantage that the HTML source does not need to be updated each time a new ARL is added.

◆ Note

The content converter feature is usable only by customers of the Akamai Freeflow Network. In addition, the features required to configure this option are available only on the BIG-IP HA and Enterprise software versions.

Creating a content converter gateway

Configuring a content converter consists of two tasks. First, configure the Akamai on-the-fly conversion software for your network. Second, create the content converter gateway using the **proxy** command. (If the software is not configured first, the attempt to create a proxy will fail.)

To configure the on-the-fly conversion software

1. On the BIG-IP, bring up the Akamai configuration file **/config/akamai.conf** in an editor like **vi** or **pico**.
2. Under the heading **[CpCode]** you will find the text **default=XXXXX**. Replace the **X**s with the CP code provided by your Akamai Integration Consultant. (If contacting your consultant, specify that you are using the BIG-IP on-the-fly akamaizer based on Akamai's 1.0 source code.) Example:

   ```
   default=773
   ```

3. Under the heading **[Serial Number]** you will find the text **staticSerialNumber=XXXXX**. Replace the **X**s with the static serial number provided by your Akamai Integration Consultant. Example:

   ```
   staticSerialNumber=1025
   ```

   Note: *This value needs to be set only if the algorithm under* ***[Serial Number]*** *is set to* ***static****, as it is in the default file. If you choose to set the algorithm to* ***deterministicHash*** *or* ***deterministicHashBounded****, the static serial number is not applicable. If you are unsure what method to select, contact your Akamai Integration Consultant.*

4. Under the heading **[URLMetaData]** you will find the text **httpGetDomains=XXXXX**. Replace the **X**s with domain name of the content to be converted. Example:

   ```
   httpGetDomains=www.f5.com
   ```

5. Save and exit the file.

To create a content converter proxy using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the Proxy Type box, check the **Akamaize** check box.
4. Configure the attributes you want to use with the proxy.
5. Click **Done**.

To configure the content converter proxy at the command line

Use the following command-line syntax to create a content converter proxy:

```
b proxy <ip>:<service> [unit <unit_id>] target server|virtual <ip>:<service> akamaize
    enable
```

For example, from the command line you can create a proxy that looks like this:

```
b proxy 10.1.1.1:80 unit 1 target virtual 20.1.1.1:80 akamaize enable
```

When the content converter proxy is written in the **/config/bigip.conf** file, it looks like the example in Figure 4.47.

```
proxy 10.1.1.1:http unit 1 {
    target virtual 20.1.1.1:http
    akamaize enable
}
```

Figure 4.47 *An example content converter proxy configuration*

Additional proxy tasks

In addition to the proxy configuration options described in the *Proxies* section of this guide, you can perform the following tasks:

- Disable or delete an SSL or content converter proxy
- Disable or delete any VLAN that is mapped to a proxy
- Display proxy configuration information

The following three sections describe these tasks.

To disable or delete a proxy from the command line

You can disable or delete a proxy with the following syntax:

```
b proxy <ip>:<service> disable
b proxy <ip>:<service> delete
```

For example, if you want to disable the SSL proxy **209.100.19.22:443**, you would type the following command:

```
b proxy 209.100.19.22:443 disable
```

If you want to delete the SSL proxy **209.100.19.22:443**, you would type the following command:

```
b proxy 209.100.19.22:443 delete
```

To disable or delete a VLAN for a proxy from the command line

A proxy is mapped by default to all VLANs on the BIG-IP. To disable or delete any VLANs to which you do not want the proxy to be mapped, use the following syntax:

```
b proxy vlans <vlan_name> disable
b proxy vlans <vlan_name> delete
```

To display configuration information for a proxy from the command line

Use the following syntax to view configuration information for the specified proxy:

```
b proxy <ip>:<service> show
```

For example, if you want to view configuration information for the SSL proxy **209.100.19.22:443**, type the following command:

```
b proxy 209.100.19.22:443 show
```

Nodes

Nodes are the network devices to which the BIG-IP passes traffic. A network device becomes a node when it is added as a member to a load balancing pool. You can display information about nodes and set properties for nodes.

The attributes you can configure for a node are listed in Table 4.21.

Node Attributes	Description
Enable/Disable nodes	You can enable or disable nodes independent of a load balancing pool.
Set node up/down	You can set a node to **up** or **down**.
Connection limit	You can place a connection limit on a node.
Associate a node with a monitor	You can associate a health monitor with a node, creating an instance of that monitor.
Add a node as a member of a pool	You can add a node to a pool as a member. This allows you to use the load balancing and persistence methods defined in the pool to control connections handled by the node.

Table 4.21 *The attributes you can configure for a node.*

To enable and disable nodes and node addresses

A node must be enabled in order to accept traffic. When a node is disabled, it allows existing connections to time out or end normally and accept new connections only if they belong to an existing persistence session. (In this way a disabled node differs from a node that is set **down**. The **down** node allows existing connections to time out, but accepts no new connections.)

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

```
b node 192.168.21.1 enable
```

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

```
b node 192.168.21.1 disable
```

To mark nodes and node ports up or down

A node must be marked **up** in order to accept traffic. When a node is marked **down** it allows existing connections to time out but accepts no new connections.

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

```
b node 192.168.21.1 down
```

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

```
b node 192.168.21.1 up
```

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

```
b node 192.168.21.1:80 down
```

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

```
b node 192.168.21.1:80 up
```

To set connection limits for nodes

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

```
b node <node_ip>[:<service>][...<node_ip>[:<service>]] limit <max conn>
```

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

```
b node 192.168.21.1:80 limit 0
```

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

```
b node 192.168.21.1 192.168.21.1 192.168.21.1 limit 100
```

To remove a connection limit, set the <**max conn**> variable to **0** (zero).

To associate a health monitor with a node

Use the following command to associate a health monitor with a node:

```
node <node> monitor use <monitor>
```

A monitor can be placed on multiple nodes and a node can have multiple monitors placed on it. To place a monitor on multiple nodes:

```
node <node_list> monitor use <monitor>
```

To place multiple monitors on a node:

```
node <node> monitor use <monitor1> and <monitor2>...
```

For more information on using the **node** command with health monitors, refer to *Health monitors*, on page 4-137.

To display status of all nodes

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

```
b node show
```

The report shows the following information:

- Current number of connections
- Total number of connections made to the node since last boot
- Maximum number of concurrent connections since the last boot
- Concurrent connection limit on the node
- The total number of connections made to the node since last boot
- Total number of inbound and outbound packets and bits

Figure 4.48 shows the output of this command.

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

Figure 4.48 *Node status and statistics*

To display the status of individual nodes and node addresses

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

```
b node 192.168.21.1 show
```

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

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

```
b node 192.168.21.1:80 show
```

To reset statistics for a node

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

```
b node [<node_ip>:<service>] stats reset
```

To add a node as a member of a pool

You can add a node as a member of a load balancing pool. For detailed information about how to do this, see *Member specification*, on page 4-5.

Services

Services are the standard Internet applications supported by BIG-IP, such as **HTTP**, **HTTPs**, **FTP**, and **POP3.** Each service is known by its name and also by its well-known or reserved port number such as **80** or **443**. (Specifically, a service is any valid service name in the **/etc/services** file or any valid port number between **0** and **65535**.) The **bigpipe service** command allows you to enable and disable network traffic on services, and also set connection limits and timeouts. You can use the service name or the port number for the **<service>** parameter. Note that the settings you define with this command control the service for all virtual servers that use it. By default, access to all services is disabled.

◆ Tip

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

Table 4.22 lists the attributes you can configure for a service.

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

Table 4.22 *The attributes you can configure for a service.*

To allow access to services using the Configuration utility

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

To allow access to services from the command line

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

```
b service <service>...<service> <protocol> [tcp|udp] enable
```

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

```
b service 80 23 443 tcp enable
```

To set connection limits on services

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

```
b service <service> [...<service>] limit <max conn>
```

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

```
b service <service> [...<service>] limit 0
```

To enable or disable TCP for services

You can enable or disable TCP for specific services. The default setting for all services is enabled. Use the following syntax to disable TCP for one or more services:

```
b service <service> [...<service>] tcp disable
```

To re-enable TCP, use this syntax:

```
b service <service> [...<service>] tcp enable
```

To enable or disable UDP for services

You can enable or disable UDP for specific services. The default setting for all services is disabled. Use the following syntax to enable UDP for one or more services:

```
b service <service> [...<service>] udp enable
```

To disable UDP, use this syntax:

```
b service <service> [...<service>] udp disable
```

To set the idle connection timeout for TCP traffic

To set the TCP timeout on one or more services, where the <**seconds**> parameter is the number of seconds before an idle connection is dropped, use the following syntax:

```
b service <service> [<service>...] timeout tcp <seconds>
```

For example, the following command sets the TCP timeout to **300** seconds for port 53:

```
b service 53 timeout tcp 300
```

To turn off TCP timeout for a service, use the above command, setting the <**seconds**> parameter to zero:

```
b service 53 timeout tcp 0
```

To set the idle connection timeout for UDP traffic

To set the UDP timeout on one or more services, where the <**seconds**> parameter is the number of seconds before an idle connection is dropped, use the following syntax:

```
b service <service> [<service>...] timeout udp <seconds>
```

For example, the following command sets the UDP timeout to **300** seconds for port 53:

```
b service 53 timeout udp 300
```

To turn off UDP timeout for a service, use the above command, setting the <**seconds**> parameter to zero:

```
b service 53 timeout udp 0
```

To display service settings

Use the following command to display the settings for all services:

```
b service show
```

Use the following syntax to display the settings for a specific service of services:

```
b service <service> [...<service>] show
```

For example, the command **b service http show** displays the output shown in Figure 4.49.

```
SERVICE 80 http tcp enabled timeout 1005 udp disabled timeout 60
           (cur, max, limit, tot, reaped) = (0, 0, 0, 0, 0)
           (pckts,bits) in = (0, 0), out = (0, 0)
```

Figure 4.49 *Sample output of the* ***bigpipe service show*** *command*

Address translation: SNATs, NATs, and IP forwarding

The BIG-IP uses address translation and forwarding in various ways to make nodes accessible that would otherwise be hidden on its internal VLAN.

- A virtual server translates the destination address of an inbound packet from its own address (the virtual server's) to the address of the node to which it load balances the packet. It then translates the origin address of the reply back to its own address so the originating host will not try to address the member node directly. This translation is basic to the way the virtual server works in most configurations and it is enabled by default.
- You can configure a SNAT (Secure Network Address Translation) or NAT (Network Address Translation) to give a node that is a member of a load balancing pool a routable address as an origin address for purposes of generating its own outbound traffic. A SNAT can be configured manually, or automatically using the SNAT auto-map feature.
- You can configure a forwarding virtual server to expose selected nodes to the external network.
- You can configure IP forwarding globally to expose all internal nodes to the external network

For more information on enabling address translation for virtual servers, refer to *Virtual servers*, on page 4-69. The following sections describe how to configure SNATs, NATs, and IP forwarding.

SNATs

A ***secure network address translation*** (SNAT) provides a routable alias IP address that a node can use as its source IP address when making connections to clients on the external network. Unlike a ***network translation address*** (NAT), a SNAT does not accept inbound traffic, and this is where its security lies. When you define a SNAT, you can use it in any of the following ways:

- Assign a single SNAT address to a single node
- Assign a single SNAT address to multiple nodes
- Enable a SNAT for a VLAN

Note that a SNAT address does not necessarily have to be unique; for example, it can match the IP address of a virtual server.

The attributes you can configure for a SNAT are shown in Table 4.23.

Attributes	Description
Global SNAT properties	Before you configure a SNAT, you can configure global properties for all SNATs on the BIG-IP. Configuring global properties for a SNAT is optional.
Manual SNAT mapping	You can define a specific translation address to be mapped to an individual host.
SNAT automapping	You can configure BIG-IP to automatically map a translation address.

Table 4.23 *The attributes you can configure for a SNAT*

Setting SNAT global properties

The SNAT feature supports three global properties that apply to all SNAT addresses:

- **Connection limits**
  The connection limit applies to each node that uses a SNAT.
- **TCP idle connection timeout**
  This timer defines the number of seconds that TCP connections initiated using a SNAT address are allowed to remain idle before being automatically disconnected.
- **UDP idle connection timeout**
  This timer defines the number of seconds that UDP connections initiated using a SNAT address are allowed to remain idle before being automatically disconnected. This value should not be set to **0**.

To configure SNAT global properties using the Configuration utility

1. In the navigation pane, click **SNATs**.
   The SNATs screen opens.
2. In the **Connection Limit** box, type the maximum number of connections you want to allow for each node using a SNAT.
3. To turn connection limits off, set the limit to **0**.
4. In the **TCP Idle Timeout** box, type the number of seconds that TCP connections initiated by a node using a SNAT are allowed to remain idle.
5. In the **UDP Idle Timeout** box, type the number of seconds that UDP connections initiated by a node using a SNAT are allowed to remain idle. This value should not be set to **0**.
6. Click the **Apply** button.

To configure SNAT global properties from the command line

Configuring global properties for a SNAT requires that you enter three **bigpipe** commands. The following command sets the maximum number of connections you want to allow for each node using a SNAT.

```
b snat limit <value>
```

The following commands set the TCP and UDP idle connection timeouts:

```
b snat timeout tcp <seconds>
b snat timeout udp <seconds>
```

When adding a default SNAT for an active-active configuration, see *Adding automapped SNATs for active-active configurations*, on page 4-129.

Configuring a SNAT manually

Once you have configured the SNAT global properties, you can manually configure SNAT address mappings. When you map a SNAT manually, you specify a particular translation IP address that you want the BIG-IP to assign from any of the following:

- One or more specified node addresses
- One or more VLANs
- A combination of specified node addresses and VLANs
- All node addresses (known as a default SNAT)

Note that a SNAT address does not necessarily have to be unique; for example, it can match the IP address of a virtual server. A SNAT address cannot match an address already in use by a NAT or another SNAT address.

The following sections describe how to add a default SNAT and how to add a SNAT manually for individual node addresses, VLANs, or a combination of both.

Adding a default SNAT manually

If you do not want to configure a SNAT for each individual node, you can manually create a default SNAT. When you add a default SNAT, you are directing the BIG-IP to map every node on the internal network to a default translation address.

◆ Note

The following procedures do not apply to active-active configurations. For information on how to add a default SNAT for an active-active configuration, see Adding automapped SNATs for active-active configurations, on page 4-129.

To add a default SNAT manually using the Configuration utility

1. In the navigation pane, click **NATs**.
   The NATs screen displays.
2. Click the SNATs tab.
3. Click the **Add Default** button.
   The Add Default SNAT screen opens.
4. In the **Translation Address** field, select the **IP** button, and type the IP address that you want BIG-IP to assign as a translation address.
5. Click **Done**.

To add a default SNAT manually from the command line

Use the following syntax to manually define the default SNAT. If you use the netmask parameter and it is different from the external interface default netmask, the command sets the netmask and derives the broadcast address.

```
b snat map default to <snat_ip> \
[vlan <vlan_name> disable|enable] \
[netmask <ip>]
```

Adding a SNAT for individual node addresses and VLANs

If you do not want to add a default SNAT, you can add a SNAT for any individual node address or VLAN. The following procedures describe how to manually add a SNAT.

To manually add a SNATusing the Configuration utility

The Configuration utility allows you to define one SNAT for one or more original IP addresses, where the original IP address can be either a specific node address or a VLAN name.

1. In the navigation pane, click **NATs**.
   The NATs screen displays.
2. Click the SNATs tab.
3. Click the **Add** button.
   The Add SNAT screen opens.
4. In the **Translation Address** field, select the **IP** button, and type the IP address that you want BIG-IP to assign as a translation address.
5. Type each node's IP address into the **Original Address:** box and move the address to the **Current List:** box, using the right arrows (>>). Also, verify that the option **choose** appears in the **VLAN** box.
6. If you want to map the translation address from a VLAN, select the VLAN name from the **VLAN** box and move the selection to the **Current List:** box, using the right arrows (>>).
7. Click **Done**.

To add a manual SNAT from the command line

The **bigpipe snat** command defines one SNAT for one or more original IP addresses, where the original IP address can be either a specific node address or a VLAN name. To manually add a SNAT using the **bigpipe snat** command, use the following syntax.

```
b snat map <orig_ip>... to <snat_ip>
```

For example, to define a SNAT for two specific nodes:

```
b snat map 192.168.75.50 192.168.75.51 to 192.168.100.10
```

To define a SNAT for two internal VLANs:

```
b snat map internal1 internal2 to 192.168.102.11
```

To define a SNAT for both a node address and a VLAN:

```
b snat map 192.168.75.50 internal2 to 192.168.100.12
```

To create individual SNAT addresses

Use the following command-line command-line syntaxsyntax to create a SNAT mapping:

```
b snat map <orig_ip> [...<orig_ip>] to \
    <snat_ip> [vlan <vlan_name> disable | enable] [unit <unit ID>] [netmask <ip>]
```

If the netmask is different from the external interface default netmask, the command sets the netmask and derives the broadcast address.

Configuring SNAT automapping

BIG-IP includes a feature called SNAT automapping. When you map a SNAT automatically, rather than manually, you enable the BIG-IP to choose the translation IP address. You also enable the BIG-IP to map that translation address from any of the following:

- One or more specified node address
- One or more VLANs
- A combination of specific node addresses and VLANs
- All node addresses (known as a default SNAT)

SNAT automapping eliminates the need for you to specifically define an IP address as the translation address.

The SNAT automapping feature is useful in the following cases:

- Where there is a need to ensure that outbound traffic returning through ISPs or NAT-less firewalls returns through the same ISP or firewall.
- Where a traditional single SNAT address would quickly exhaust the number of ephemeral ports available. As long as there is more than one eligible self IP address, SNAT automapping can increase the number of simultaneous connections possible by using the same ephemeral port on multiple addresses.

- When the *equivalent* of a default SNAT, that is, a SNAT that continues to work in the event of a failure in one BIG-IP, is required for BIG-IP units in active-active mode. (The conventional default SNAT does not work in active-active mode.)

Adding an automapped default SNAT

The BIG-IP allows you to take advantage of the SNAT automapping feature when adding a default SNAT. When you add a default SNAT, you are enabling the BIG-IP to map every node on the internal network to a default translation address. With the automapping feature, you do not need to define a specific translation address to which all nodes on the network will be mapped.

To add the automapped default SNAT using the Configuration utility

1. In the navigation pane, click **NATs**.
   The NATs screen displays.
2. Click the SNATs tab.
3. Click the **Add Default** button.
   The Add Default SNAT screen opens.
4. Click the **Automap** button.
5. Click **Done**.

To add the automapped default SNAT from the command line

To add a default SNAT using the automapping feature, type the **bigpipe snat** command as follows:

```
b snat map default to auto
```

◆ Note

A default SNAT cannot be added for an active-active configuration. For more information, see ***Adding automapped SNATs for active-active configurations****, on page 4-129.*

Adding automapped SNATs for standard (active-standby) configurations

When enabling SNAT automapping for VLANs, the BIG-IP handles the SNATs in the following ways:

- If you create a SNAT on an internal VLAN, a SNAT is performed on any connection made from that VLAN.
- If you enable **snat automap** on a single self IP address, the translation address is that self IP address.

- If you enable **snat automap** on more than one self IP address, (implying more than one IP network), the following rules apply:
  - If the connection is handled by a non-forwarding virtual server, the translation address is the self IP address that matches the IP network of the node selected by load balancing.
  - If the connection is handled by a forwarding virtual server or no virtual server, the translation address is the self IP address that matches the IP network of the next hop to the destination.
  - If there are no self addresses that match the IP network of the node or the next hop, any self IP address on the VLAN is eligible.

To add a SNAT using the automapping feature, you must complete two procedures:

- Enable the **snat automap** attribute on any self IP addresses.
- Add the SNAT, specifying the **Automap** feature.

The following sections explain these procedures.

To enable the snat automap attribute on a self IP address from the command line

When you enable automapping to add a SNAT, the translation address that the BIG-IP maps to an individual node or a VLAN is the self IP address. Thus, prior to enabling automapping for the node or VLAN, you must enable the **snat automap** attribute on the self IP address. This is done from the command line, using the following syntax:

```
b self <self IP address> snat automap enable
```

For example, if you have the two self IP addresses **192.168.217.14** and **192.168.217.15**, the following commands enable the **snat automap** attribute on those self IP addresses:

```
b self 192.168.217.14 snat automap enable
b self 192.168.217.15 snat automap enable
```

Later, when you add a SNAT using automapping, the BIG-IP maps either of those self IP addresses to the original node (or VLAN) that you specify.

As another example, the following command enables the **snat automap** attribute on the self IP address **10.0.0.1**, for the VLAN named **external**:

```
b self 10.0.0.1 vlan external snat automap enable
```

For more information, see *To add an automapped SNAT from the command line*, on page 4-129.

To add an automapped SNAT using the Configuration utility

The Configuration utility allows you to define one SNAT for one or more original IP addresses, where the original IP address can be either a specific node address or a VLAN name.

1. In the navigation pane, click **NATs**.
   The **NAT**s screen displays.

2. Click the SNATs tab.
3. Click the **Add** button.
   The Add SNAT screen opens.
4. In the Translation Address dialog area, click the **Automap** button.
5. If you want to map the translation address from one or more specific nodes, enter each node's IP address into the **Original Address:** box and move the address to the **Current List:** box, using the right arrows (>>). Also, verify that the option **choose** appears in the **VLAN** box.
6. If you want to map the translation address to a VLAN, select the VLAN name from the **VLAN** box and move the selection to the **Current List:** field, using the right arrows (>>).
7. Click **Done**.

To add an automapped SNAT from the command line

The **bigpipe snat** command defines one SNAT for one or more original IP addresses, where the original IP address can be either a specific node address, or a VLAN name.

For example, to define an automapped SNAT for two individual node addresses:

```
b snat map 10.1.1.1 10.1.1.2 to auto
```

In the preceding example, the translation address to which the nodes **10.1.1.1** and **10.1.1.2** will be mapped is the self IP address, assuming that you enabled the **snat automap** attribute on that self IP address prior to using the **bigipipe snat** command. For more information, see *To enable the snat automap attribute on a self IP address from the command line,* on page 4-128.

To define an automapped SNAT for a VLAN named **internal**:

```
b snat map internal to auto
```

To define an automapped SNAT for both a node address and a VLAN:

```
b snat map 192.168.75.50 internal2 to auto
```

◆ Note

*When adding automapped SNATs, you must also enable the **snat automap** attribute on the self IP address that the BIG-IP will use as the translation address. For more information, see **To enable the snat automap attribute on a self IP address from the command line**, on page 4-128.*

Adding automapped SNATs for active-active configurations

In the case where you want to add a default SNAT for an active-active configuration, you cannot create the standard default SNAT described earlier in this section. Instead, you must create the *equivalent* of a default SNAT.

To create the equivalent of a default SNAT, it is necessary to assign each unit its own floating self IP address on the external VLAN. This is done for the same reason that separate aliases are assigned to the internal network as part of routine active-active setup. (See *Configuring an active-active system*, on page 6-11.) Because you already have a floating self IP address for the external interface that is configured as belonging to unit one on unit one and unit two on unit two, use the following procedure to create two unit-specific IP aliases is as follows.

To create two unit-specific SNATs

1. On unit one, ensure that two floating self IP addresses are configured for unit one. For example:

```
b self 11.11.11.3 vlan internal unit 1 floating enable

b self 172.16.16.3 vlan external unit 1 floating enable
```

2. Also on unit one, ensure that two floating self IP addresses are configured for unit two. For example:

```
b self 11.11.11.4 vlan internal unit 2 floating enable

b self 172.16.16.4 vlan external unit 2 floating enable
```

3. Ensure that unit two has all of these self IP addresses by using the **config sync** command to synchronize the changes to unit two:

```
b config sync all
```

4. Set up SNAT automapping as you would for an active/standby system, but enable both external aliases:

```
b self 172.16.16.3 vlan external snat automap enable

b self 172.16.16.4 vlan external snat automap enable

b snat map internal to auto
```

ISPs and NAT-less firewalls

BIG-IP handles ISPs and NAT-less firewalls in the following manner:

- If multiple external interfaces are available, the inside addresses of the firewalls in the load balancing pool may each be connected to different interfaces and assigned to different VLANs.
- A SNAT is then enabled on each VLAN.
- A SNAT must also be enabled on the internal VLAN.

For example, if the internal VLAN is named **internal** and the external VLANs are named **external1** and **external2**, you would type the following commands:

```
b snat internal to auto
b snat external1 to auto
b snat external2 to auto
```

- If multiple external interfaces are not available, the ISP routers or firewalls are assigned to different IP networks. This will already be the case for ISPs.
- For firewalls, the separate IP address ranges must be established on the inside and outside interfaces of each firewall. The separate networks are then assigned separate self addresses, for example, **10.0.0.1** and **11.0.0.1**.

Thus, if the internal and external VLANs are named **internal** and **external**, you would type the following commands:

```
b self 10.0.0.1 vlan external snat automap enable
b self 11.0.0.1 vlan external snat automap enable
b snat internal to auto
```

Disabling SNATs and NATs for a pool

When configuring a pool, you can specifically disable SNAT or NAT translations on any connections that use that pool. By default, this setting is enabled. For information on how to disable SNAT and NAT connections for a pool, see *Disabling SNAT and NAT connections*, on page 4-45.

Disabling ARP requests

By default, the BIG-IP responds to ARP requests for the SNAT address and sends a gratuitous ARP request for router table update. If you want to disable the SNAT address for ARP requests, you must specify **arp disable**.

Additional SNAT configuration options

The following procedures allow you to further configure SNATs.

To delete SNAT addresses

The following syntax deletes a specific SNAT:

```
b snat <snat_ip> | default delete
```

To show SNAT mappings

The following **bigpipe** command shows mappings:

```
b snat [<snat_ip> ...] show
b snat default show
```

The value of the **<snat_ip>** variable can be either the translated or the original IP address of the SNAT, or a SNAT-enabled VLAN name.

The following command shows the current SNAT connections:

```
b snat [<snat_ip> ...] dump [ verbose ]
b snat default dump [ verbose ]
```

The optional **verbose** keyword provides more detailed output.

The following command prints the global SNAT settings:

```
b snat globals show
```

To enable mirroring for redundant systems

The following example sets SNAT mirroring for all SNAT connections originating at **192.168.225.100**:

```
b snat 192.168.225.100 mirror enable
```

To clear statistics

You can reset statistics by node address, SNAT address, or VLAN name. Use the following syntax to clear all statistics for one or more nodes:

```
b snat <node_ip> ... stats reset
```

Use the following syntax to clear all statistics for one or more SNAT addresses:

```
b snat <snat_ip> ... stats reset
```

Use the following command to reset the statistics to zero for the default:

```
b snat default stats reset
```

NATs

A ***network translation address*** (NAT) provides a routable alias IP address that a node can use as its source IP address when making or receiving connections to clients on the external network. (This distinguishes it from a SNAT, which can make outbound connections but refuses inbound connections.) You can configure a unique NAT for each node address included in a virtual server mapping.

◆ Note

Note that NATs do not support port translation, and are not appropriate for protocols that embed IP addresses in the packet, such as FTP, NT Domain or CORBA IIOP. You cannot define any NATs if you configure a default SNAT.

Table 4.24 shows the attributes you can configure for a NAT.

NAT Attributes	Description
Original address	The original address is the node IP address of a host that you want to be able to connect to through the NAT.
Translated address	The translated address is an IP address that is routable on the external network of the BIG-IP. This IP address is the NAT address.
Disabled VLAN list	VLANs to which the NAT is not to be mapped can be explicitly disabled, as when there is more than one internal VLAN.
Unit ID	You can specify a unit ID for a NAT if the BIG-IP is configured to run in active-active mode.

Table 4.24 *The attributes you can configure for a NAT*

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

Using network address translation resolves this problem. Network address translations (NATs) assign to a particular node a routable IP address that the node can use as its source IP address when connecting to servers on the BIG-IP external interface. You can use the NAT IP address to connect directly to the node through the BIG-IP, rather than having the BIG-IP send you to a random node according to the load balancing mode.

◆ Note

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

Defining a network address translation (NAT)

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

To configure a NAT using the Configuration utility

1. In the navigation pane, click **NATs**.
   The NATs screen opens.
2. Click the **Add** button.
   The Add NAT screen opens.
3. In the Add NAT screen, fill in the fields to configure the NAT. For additional information configuring a NAT, click the **Help** button.

To configure a NAT from the command line

A NAT definition maps the IP address of a node <**orig_addr**> to a routable address on the external interface <**trans_addr**>. Use the following syntax to define a NAT:

```
b nat <orig_addr> to <trans_addr> [vlans <vlan_list> disable | enable] [unit <unit ID>]
```

The **vlans <vlan_list>** parameter is used to disable the specified VLANs for translation. By default, all VLANs are enabled.

Use the **unit <unit ID>** parameter to specify the BIG-IP to which this NAT applies in an active-active redundant system.

The following example shows a NAT definition:

```
b nat 10.10.10.10 to 10.12.10.10
```

To delete NATs

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

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

To display status of NATs

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

```
b nat show
```

Use the following syntax to display the status of one or more selected NATs (see Figure 4.50).

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

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

Figure 4.50 *Output when you display the status of a NAT*

To reset statistics for a NAT

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

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

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

```
b nat stats reset
```

Disabling SNATs and NATs for a pool

When configuring a pool, you can specifically disable any SNAT or NAT connections that use that pool. By default, this setting is enabled. For information on how to disable SNAT and NAT connections for a pool, see *Disabling SNAT and NAT connections*, on page 4-45.

Disabling ARP requests

By default, the BIG-IP responds to ARP requests for the NAT address and sends a gratuitous ARP request for router table update. If you want to disable the NAT address for ARP requests, you must specify **arp disable**.

Additional restrictions

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

- The IP address defined in the **<orig_addr>** parameter must be routable to a specific server behind the BIG-IP.

- You must delete a NAT before you can redefine it.
- The interface for a NAT can only be configured when the NAT is first defined.

IP forwarding

IP forwarding is an alternate way of allowing nodes to initiate or receive direct connections from the BIG-IP external network. IP forwarding directly exposes all of the node IP addresses to the external network, making them routable on that network. If your network uses the NT Domain or CORBA IIOP protocols, IP forwarding is an option for direct access to nodes.

◆ Tip

Use of SNATs and NATs, as well as forwarding pools and forwarding virtual servers, is preferable to global IP forwarding. For more information on forwarding pools and forwarding virtual servers, see ***Forwarding pools****, on page 4-47 and* ***Forwarding virtual servers****, on page 4-76.*

IP forwarding is a global setting that exposes the IP address of all internal nodes to the BIG-IP external network, and clients can use it as a standard routable address. When you turn IP forwarding on, the BIG-IP acts as a router when it receives connection requests for node addresses. You can use the IP filter feature to implement a layer of security that can help protect your nodes.

Table 4.25 shows options associated with IP forwarding.

Option	Description
Enable IP forwarding globally	You can enable IP forwarding globally for the BIG-IP, either with the Configuration utility or by turning on the **sysctl** variable **net.inet.ip.forwarding**. To protect your nodes with this feature, we recommend that you use IP filters, which add a layer of security.
Address routing issues	If you enable IP forwarding, you need to route packets to the node addresses through the BIG-IP.
Configure the forwarding attribute for a pool	Instead of enabling IP forwarding globally or creating a forwarding virtual server, you can create a pool with no members that forwards traffic instead of load balancing it. For more information, see *Forwarding pools*, on page 4-47.
Enable IP forwarding for a virtual server	Instead of enabling IP forwarding globally, you can create a special virtual server with IP forwarding enabled. For information on creating a forwarding virtual server, see *Forwarding virtual servers*, on page 4-76.

Table 4.25 *The attributes you can configure for IP forwarding*

Enabling IP forwarding globally

IP forwarding is a global property of the BIG-IP system. To set up IP forwarding globally, you need to complete two tasks:

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

To set global IP forwarding using the Configuration utility

1. In the navigation pane, click **System**.
   The Network Map screen opens.
2. Click the Advanced Properties tab.
   The Advanced Properties screen opens.
3. Check the **Allow IP Forwarding** box.
4. Click **Apply**.

To set global IP forwarding from the command line

Use the **bigpipe global ip_forwarding** command to set the variable. The default setting for the variable is **disabled**. You should change the setting to **enabled**:

```
b global ip_forwarding enabled
```

Addressing routing issues for IP forwarding

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

Configuring the forwarding attribute for a pool

You can configure IP forwarding so that it is done by a pool, rather than globally by the BIG-IP or by an individual virtual server. For more information, see *Forwarding pools*, on page 4-47.

Enabling IP forwarding for a virtual server

You can configure IP forwarding so that it is done by a virtual server, rather than globally by the BIG-IP or by a specific pool. For more information, see *Forwarding virtual servers*, on page 4-76.

Health monitors

Health monitors verify connections and services on nodes that are members of load balancing pools. The monitor checks the node at a set interval. If the node does not respond within a specified timeout period, the node is marked **down** and traffic is no longer directed to it.

By default, an **icmp** (Internet Control Message Protocol) monitor is associated with every node that is a member of a load balancing pool. This monitor is of the simplest type, checking only the node address and checking only for a ping response. To change the interval and timeout values of this default check, or to check specific services on a node, you need to configure a custom monitor or monitors to add to the default monitor. The BIG-IP provides a variety of service-specific monitors in template form. Some of these monitors are usable as is (assuming their default values are acceptable) and may be put in service simply by associating them with the nodes to be monitored. In most cases, however, the template is used purely as a template for configuring custom monitors. Configuring custom monitors and placing them in service is a three-step process:

- Selecting the template
- Configuring the monitor from the template
- Associating the monitor with the node or nodes

For example, for the default **icmp** monitor, we selected the **icmp** monitor template, as shown in Figure 4.51.

```
monitor type icmp {
    interval 5
    timeout 16
    dest *
}
```

Figure 4.51 *The **icmp** monitor template*

The **icmp** monitor template has three attributes, **interval**, **timeout**, and **dest**, each with a default value. (All monitor templates have these three basic attributes. Other monitor templates have additional attributes as required by the service type.) These attributes are inherited by the custom monitor when it is configured and can be left at their default values or assigned new values as required.

For the default monitor, template **icmp** is used as is, that is, as monitor **icmp** with its default attribute values. To change any of these default values, you need to create a custom monitor based upon **icmp**, for example, **my_icmp**. Only the values that are actually to be changed would need to be specified in the definition of the custom monitor. Therefore, if you wanted to change the timeout values only, you define the custom monitor as follows:

```
b monitor my_icmp '{ use icmp timeout 20 }'
```

This creates a new monitor in **/config/bigip.conf**, as shown in Figure 4.52. You can display this monitor using the command **b monitor my_icmp show**.

```
monitor my_icmp{
    #type icmp
    "icmp"
    interval 5
    timeout 20
    }
```

Figure 4.52 *Custom **icmp** monitor*

Once the custom monitor exists, you associate it with a node or nodes using the Configuration utility or the **bigpipe node** command as follows.

```
b node 11.11.11.1 11.11.11.2 11.11.11.3 monitor use my_icmp
```

◆ Note

*The nodes are identified by IP address only. **icmp** can ping addresses only, not specific ports on addresses. This creates three instances of monitor **my_icmp**, one for each address. You can display the instances using the command **b node monitor my_icmp show**.*

```
+- NODE  ADDRESS 11.11.11.1   UP
|    |
|    +- icmp
|       11.11.11.1 up enabled
|
+- NODE  ADDRESS 11.11.11.2   UP
|    |
|    +- icmp
|       11.11.11.2 up enabled
|
+- NODE ADDRESS 11.11.11.3  UP
     |
     +- icmp
        11.11.11.3 up enabled
```

Figure 4.53 *Output for the command **b node monitor show***

Note that each instance takes as its destination the same node it is associated with. This is because the **dest** value in **my_icmp** was left at the default *****, which tells the instance to use the associated node as its destination. Assigning a specific address to **dest**, such as **11.11.11.1**, would cause the monitor to verify all three addresses by checking that one address, making **11.11.11.2** and **11.11.11.3** dependent on **11.11.11.1**.

Selecting the monitor template

Selecting a template is straightforward. Like **icmp**, each of the templates has a **type** based on the type of service it checks, for example, **http**, **https**, **ftp**, **pop3**, and takes that type as its name. (Exceptions are port-specific templates, like **https_443**, and the **external** template, which calls a user-supplied program.) To select a template, simply select the one that corresponds in name and/or type to the service you want to check. If more than one service is to be checked, for example **http** and **https**, more than one monitor can be placed on the node. (This creates a rule, namely that the node will not be considered up unless both monitors run successful checks.) You may not want to check all services available on a node specifically. If you want to verify only that the destination IP address is alive, or that the path to it through a transparent node is alive, use one of the simple templates, **icmp** or **tcp_echo**. If you want to verify TCP only, use the monitor template **tcp**.

All monitor templates are contained in the read-only file **/etc/base_monitors.conf**. The following sections describe each of the monitor templates, its function, and the information required to configure a monitor from it. The templates are divided into three groups based on the types of monitors they support: simple monitors, ECV (Extended Content Verification) monitors, and EAV (Extended Application Verification) monitors. Also described are the port-specific monitor templates, which are derived from the other types.

Working with templates for simple monitors

Simple monitors are those that check node addresses only and verify simple connections only. Templates for these monitors are **icmp** and **tcp_echo**.

◆ Note

*The templates **icmp** and **tcp_echo** are both usable as is, that is, they may be associated with nodes. It is important to understand, however, that using a template as is means that you are using the default attribute values. To change any of these values, you have to configure a custom monitor based on the template.*

Using icmp

The **icmp** template uses Internet Control Message Protocol to make a simple node check. The check is successful if a response to an ICMP_ECHO datagram is received. **icmp** has no attributes other than the standard **interval**, **timeout**, and **dest**.

```
monitor icmp {
    #type icmp
    interval 5
    timeout 16
    dest *
    }
```

Figure 4.54 *The **icmp** monitor template*

Using tcp_echo

The **tcp_echo** template uses Transmission Control Protocol. The check is successful if a response to a TCP ECHO message is received. **tcp_echo** also supports ***transparent*** mode. In this mode, the node with which the monitor is associated is pinged through to the destination node. (For more information about transparent mode, refer to *Using transparent and reverse modes*, on page 4-150.)

To use **tcp_echo**, you must ensure that TCP ECHO is enabled on the nodes being monitored.

```
monitor tcp_echo  {
    #type tcp_echo
    interval 5
    timeout 16
    dest *
    //transparent
    }
```

Figure 4.55 *The* ***tcp_echo*** *monitor template*

Working with templates for ECV monitors

ECV monitors attempt to retrieve explicit content from nodes using **send** and **recv** statements. These include **http**, **https** and **tcp**.

◆ Note

The templates ***http****,* ***https,*** *and* ***tcp*** *are all usable as is, and you may associate them with nodes. It is important to understand, however, that using a template as is means that you are using the default attribute values. To change any of these values, you have to configure a custom monitor based on the template.*

Using tcp

The **tcp** template is for Transmission Control Protocol. A **tcp** monitor attempts to receive specific content. The check is successful when the content matches the **recv** expression. A **tcp** monitor takes a **send** string and a **recv** expression. If the **send** string is left blank, the service is considered up if a connection can be made. A blank **recv** string matches any response.

Both **transparent** and **reverse** modes are options. (For more information about transparent and reverse modes, refer to *Using transparent and reverse modes*, on page 4-150.)

```
monitor tcp  {
    #type tcp
    interval 5
    timeout 16
    dest *:*
    send ""
    recv ""
    //reverse
    //transparent
    }
```

Figure 4.56 *The* ***tcp*** *monitor template*

Using http

The **http** template is for HyperText Transfer Protocol. Like a **tcp** monitor, an **http** monitor attempts to receive specific content from a web page, and unlike a **tcp** monitor, sends a user name and password. The check is successful when the content matches the **recv** expression. An **http** monitor uses a **send** string, a **recv** expression, **username**, **password**, and optional **get**, **url**, **transparent** and **reverse** statements. (If there is no password security, use blank strings [**""**] for **username** and **password**.) The optional **get** statement replaces the send statement, automatically filling in the string **"GET"**. Thus the following two statements are equivalent:

```
send "GET/"
get "/"
```

The optional **url** statement takes the HTTP URL as a value and automatically fills in the **dest** value with the address the URL resolves to. (For more information about transparent and reverse modes, refer to *Using transparent and reverse modes*, on page 4-150.) Both **transparent** and **reverse** modes are also options. (For more information about the **get** and **url** statements, refer to *Using send, receive, url, and get statements*, on page 4-150.)

```
monitor http {
    #type http
    interval 5
    timeout 16
    dest *:*
    send "GET /"
    recv ""
    username ""
    password ""
    //get
    //url
    //reverse
    //transparent
    }
```

Figure 4.57 *The* ***http*** *monitor template*

Using https

The **https** template is for Hypertext Transfer Protocol Secure. An **https** monitor attempts to receive specific content from a web page protected by SSL security. The check is successful when the content matches the **recv** expression. An **https** monitor uses a **send** string, a **recv** expression, and a **username** and **password** (If there is no password security, use blank strings [""] for **username** and **password**.) The optional **get** statement replaces the send statement, automatically filling in the string **"GET"**. Thus, the following two statements are equivalent:

```
send "GET/"
get "/"
```

The optional **url** statement takes the HTTPS URL as a value and automatically fills in the **dest** value with the address the URL resolves to.

```
monitor https {
    #type https
    interval 5
    timeout 16
    dest *:*
    send "GET /"
    recv ""
    //get
    //url
    username ""
    password ""
    }
```

Figure 4.58 *The* ***https*** *monitor template*

Working with templates for EAV monitors

EAV monitors verify applications on the node by running those applications remotely, using an external service checker program located in the directory **/user/local/lib/pingers**. These include **ftp**, **pop3**, **smtp**, **sq.**, **nntp**, **imap**, **ldap**, and **radius**. Also included is the template **external**, which has a **run** attribute to specify a user-added external monitor.

Using ftp

The **ftp** template is for File Transfer Protocol. The monitor attempts to download a specified file to the **/var/tmp** directory. The check is successful if the file is retrieved. The **ftp** monitor takes a **get** statement, **username**, and **password**. The **get** statement takes the full path to the file as a value. The optional **url** statement may be used in place of **get**. The **url** takes the FTP

URL as a value and automatically fills in the **dest** value with the address the URL resolves to. (For more information about the **get** and **url** statements, refer to *Using send, receive, url, and get statements*, on page 4-150.)

```
monitor ftp {
    #type ftp
    interval 5
    timeout 16
    dest *:*
    username ""
    password ""
    get ""
    //url
    }
```

Figure 4.59 *The **ftp** monitor template*

Using pop3

The **pop3** template is for Post Office Protocol. The check is successful if the monitor is able to connect to the server, log in as the indicated user, and log out. The **pop3** monitor requires **username** and **password**.

```
monitor pop3 {
    #type pop3
    interval 5
    timeout 16
    dest *:*
    username ""
    password ""
    }
```

Figure 4.60 *The **pop3** monitor template*

Using smtp

The **smtp** template is for Simple Mail Transport Protocol servers. An **smtp** monitor is an extremely simple monitor that checks only that the server is up and responding to commands. The check is successful if the mail server responds to the standard SMTP **HELO** and **QUIT** commands. An **smtp** monitor requires a **domain** name**.**

```
monitor smtp {
    #type smtp
    interval 5
    timeout 16
    dest *:*
    domain ""
    }
```

Figure 4.61 *The **smtp** monitor template*

Using snmp_dca

The **snmp_dca** template is used for load balancing traffic to servers that are running an SNMP agent, such as UC Davis or Windows 2000. In addition to defining ratio weights for CPU, memory, and disk use, you can also define weights for use by users. Figure 4.62 shows the **snmp_dca** monitor template.

```
monitor type snmp_dca {
    #type snmp_dca
    interval 10
    timeout 30
    dest *:161
    agent_type "UCD"
    cpu_coefficient "1.5"
    cpu_threshold "80"
    mem_coefficient "1.0"
    mem_threshold "70"
    disk_coefficient "2.0"
    disk_threshold "90"
}
```

***Figure 4.62 snmp_dca** monitor template*

For detailed information on using the **snmp_dca** template, see *Configuring SNMP servers*, on page 4-16.

Using snmp_dca_base

Like the **snmp_dca** template, the **snmp_dca_base** template is for load balancing traffic to servers that are running an SNMP agent, such as UC Davis or Windows 2000. However, this template should be used only when you want the load balancing destination to be based solely on user data, and not CPU, memory, or disk use. Figure 4.63 shows the **snmp_dca_base** monitor template.

```
monitor type snmp_dca_base {
    #type snmp_dca_base
    interval 10
    timeout 30
    dest *:161
    }
```

***Figure 4.63 snmp_dca_base** monitor template*

For detailed information on using the **snmp_dca_base** template, see *Configuring SNMP servers*, on page 4-16.

Using nntp

The **nntp** template is for Usenet News. The check is successful if the monitor retrieves a newsgroup identification line from the server. An **nntp** monitor requires a **newsgroup** name (for example, **"alt.cars.mercedes"**) and, if necessary, **username** and **password**.

```
monitor nntp {
    #type nntp
    interval 5
    timeout 16
    dest *:*
    username ""
    password ""
    newsgroup ""
    }
```

Figure 4.64 *The **nntp** monitor template*

Using sql

The **sql** template is for service checks on SQL-based services such as Microsoft SQL Server versions 6.5 and 7.0, and also Sybase. The service checking is accomplished by performing an SQL login to the service. An executable program, **tdslogin** performs the actual login. The check is successful if the login succeeds.

An **sql** monitor requires a **database** (for example, **"server_db"**), **username**, and **password**.

```
monitor sql {
    #type sql
    interval 5
    timeout 16
    dest *:*
    username ""
    password ""
    database ""
    }
```

Figure 4.65 *The **sql** monitor template*

Using imap

The **imap** template is for Internet Message Access Protocol. The **imap** monitor is essentially a **pop3** monitor with the addition of the attribute **folder**, which takes the optional key **message_num**. The check is successful

if the specified message number is retrieved. An **imap** monitor requires **username**, **password**, and a **folder**. It also takes an optional message number, **message_num**.

```
monitor imap {
    #type imap
    interval 5
    timeout 16
    dest *:*
    username ""
    password ""
    folder ""
    /message_num ""
    }
```

Figure 4.66 *The **imap** monitor template*

Using radius

The **radius** template is for Remote Access Dial-in User Service servers. The check is successful if the server authenticates the requesting user. A **radius** monitor requires a **username**, a **password**, and a shared secret string **secret** for the code number.

◆ Note

*Servers to be checked by a **radius** monitor typically require special configuration to maintain a high level of security while also allowing for monitor authentication.*

```
monitor radius {
    #type radius
    interval 5
    timeout 16
    dest *
    username ""
    password ""
    secret ""
    }
```

Figure 4.67 *The **radius** monitor template*

Using ldap

The **ldap** template is for Lightweight Directory Access Protocol, which implements standard X.500 for e-mail directory consolidation. A check is successful if entries are returned for the **base** and **filter** specified. An **ldap** monitor requires a **username**, a **password**, and a **base** and a **filter** string. The **username** is a distinguished name, that is, an LDAP-format user name. The **base** is the starting place in the LDAP hierarchy from which to begin the query. The **filter** is an LDAP-format key of what is to be searched for.

Note

*Servers to be checked by an **imap** monitor typically require special configuration to maintain a high level of security while also allowing for monitor authentication.*

```
monitor ldap {
    #type ldap
    interval 5
    timeout 16
    dest *:*
    username ""
    password ""
    base ""
    filter ""
    }
```

***Figure 4.68** A Sample monitor template*

Using external

The **external** template is for a user-supplied monitor. An **external** monitor requires the executable name (**run**) of that monitor and any command line arguments (**args**) required.

```
monitor external {
    #type external
    interval 5
    timeout 16
    dest *:*
    run ""
    args ""
    }
```

***Figure 4.69** The external monitor template*

Configuring a monitor

The second step in creating a monitor and placing it in service is to configure the monitor from the monitor template. Configuring a monitor consists of giving it a name distinct from the monitor template name and assigning values to all attributes that are not to be left at their default values (and adding any optional attributes that are not present by default, like **reverse** or **transparent**). You can do this using the Configuration utility or at the command line using the **bigpipe monitor** command.

To configure a monitor using the Configuration utility

1. In the navigation pane, click **Monitors**.
   The Network Monitors screen opens.
2. Click the **Add** button.
   The Add Monitor screen opens.

3. In the Add Monitor screen, type in the name of your monitor (it must be different from the monitor template name), and select the monitor template you want to use.
4. Click the **Next** button and you are guided through the configuration of your monitor.
5. When you have finished configuring the monitor, click **Done**.

To configure a monitor from the command line

Use the **bigpipe monitor** command to configure the monitor at the command line. If you are defining the monitor with all attributes set to their default values, type:

```
b monitor <name> '{ use <template_name> }'
```

If you want to set one or more attributes to a new value, specify only those attributes and their values. For example, to create a **tcp_echo** monitor **my_tcp_echo** in **bigpipe** using the default values for the attributes **interval**, **timeout**, and **dest**, you would type:

```
b monitor my_tcp '{ use tcp_echo }'
```

If you are changing any of the default values, you need to specify only these changes. For example:

```
b monitor my_tcp-echo '{ use tcp_echo interval 10 timeout 20 }'
```

If you are using an optional attribute, such as **transparent**, add it to the list:

```
b monitor my_tcp-echo '{ use tcp_echo interval 10 timeout 20 transparent dest
    198.192.112.13:22}'
```

◆ Note

*If you are configuring an **snmp_dca** or **snmp_dca_base** monitor, see also **Configuring SNMP servers**, on page 4-16.*

Monitor attributes

Table 4.26 provides a summary of the monitor attributes and their definitions. For more information on the monitor templates and attributes, refer to *Selecting the monitor template*, on page 4-139.

Attribute	Definition
interval <seconds>	Ping frequency time interval in seconds.
timeout <seconds>	Ping timeout in seconds.
dest <node_addr>	Ping destination node. <node_address> Usually *:* for simple monitors, *:* for all others, causing the monitor instance to ping the address or address:port for which it is instantiated. Specifying address and/or port forces the destination to that address/port.

Table 4.26 *Monitor attributes*

Attribute	Definition
send <string>	Send string for ECV. Default **send** and **recv** values are empty (""), matching any string.
recv <string>	Receive expression for ECV. Default **send** and **recv** values are empty (""), matching any string.
get <string>	For the **http** and **https** monitors **get** replaces the **recv** statement, automatically filling in **"GET".** For the **ftp** monitor **get** can be used to specify a full path to a file. This automatically fills in **dest**.
url	For the **http** and **https**, and **ftp** monitors, **url** replaces the **recv** statement, supplies a URL and automatically fills in **dest** with the URL address.
reverse	A mode that sets the node down if the received content matches the **recv** string.
transparent	A mode that forces pinging through the node to the **dest** address for transparent nodes, such as firewalls.
run <program>	An external user-added EAV program.
args <program_args>	List of command line arguments for external program. The **args** are quoted strings set apart by spaces.
username <username>	User name for services with password security. For **ldap** this is a **distinguished**, that is, LDAP-format user name.
password <password>	Password for services with password security.
newsgroup <newsgroup>	Newsgroup, for type **nntp** EAV checking only
database <database>	Database name, for type **sql** EAV checking only.
domain <domain_name>	Domain name, for type **smtp** EAV checking only
secret	Shared secret for **radius** EAV checking only.
folder	Folder name for **imap** EAV checking only.
message_num	Optional message number for imap EAV.
base	Starting place in the LDAP hierarchy from which to begin the query, for **ldap** EAV checking only.
filter	LDAP- format key of what is to be searched for, for **ldap** EAV checking only.

Table 4.26 *Monitor attributes*

Entering string values

Except for **interval**, **timeout**, and **dest**, you should enter all attribute values as quoted strings, even if they are numeric, as in the case of code numbers.

Setting destinations

By default, all **dest** values are set to the wildcard **"*"** or **"*:*"**. This causes the monitor instance created for a node to take that node's address or address and port as its destination. An explicit **dest** value is used only to force the instance destination to a specific address and/or port which may not be that of the node. For more information about setting destinations, refer to *Associating the monitor with a node or nodes*, on page 4-154.

Using send, receive, url, and get statements

The ECV monitor templates **http, https,** and **tcp** have the attributes **send** and **recv** for the send string and receive expression, respectively.

The most common send string is **"GET /"** which simply retrieves a default HTML page for a web site. To retrieve a specific page from a web site, simply enter a fully qualified path name:

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

The receive expression is the text string the monitor looks for in the returned resource. The most common receive expressions contain a text string that would be included in a particular HTML page on your site. The text string can be regular text, HTML tags, or image names.

The sample receive expression below searches for a standard HTML tag.

```
"<HEAD>"
```

You can also use the default null **recv** value **""**. In this case, any content retrieved is considered a match. If both **send** and **recv** are left empty, only a simple connection check is performed.

For **http** and **ftp**, the special attributes **get** or **url** may be used in place of **send** and **recv** statements. The attribute **get** takes the full path to the file as a value and automatically fills in the **dest** value with the address the path resolves to. The following two statements are equivalent:

```
send "GET/"
get "/"
```

The attribute **url** takes the URL as a value and automatically fills in the **dest** value with the address the URL resolves to. The URL is then resolved to supply the **dest** address automatically. The third statement below is equivalent to the first two combined:

```
dest 198.192.112.13:22
get "/"
```

```
url "ftp://www.my_domain.com/"
```

Using transparent and reverse modes

The ECV monitors have optional keywords **transparent** and **reverse**. (The keyword **transparent** may also be used by **tcp_echo**.) The normal and default mode for a monitor is to ping the **dest** node by an unspecified route and to mark the node **up** if the test is successful. There are two other modes, transparent and reverse.

In transparent mode, the monitor is forced to ping *through* the node it is associated with, usually a firewall, to the **dest** node. (In other words, if there are two firewalls in a load balancing pool, the destination node will always be pinged through the one specified and not through the one picked by the load balancing method.) In this way, the transparent node is tested as well: if there is no response, the transparent node is marked **down**. For more information about transparent mode, refer to *Using transparent mode*, on page 4-157.

In reverse mode, the monitor marks the node **down** when the test is successful. For example, if the content on your web site home page is dynamic and changes frequently, you may want to set up a reverse ECV service check that looks for the string **"Error"**. A match for this string would mean that the web server was down. Transparent mode can also be used with **tcp_echo**.

Transparent and reverse modes cannot be used on the same monitor.

Testing SQL service checks

SQL service checks may require manual testing before being implemented in a monitor, as follows:

```
cd /usr/local/lib/pingers
./tdslogin 192.168.1.1 1433 mydata user1 mypass1
```

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

You should receive the message:

```
Login succeeded!
```

If you receive the connection refused message, verify that the IP and port are correct.

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

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

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

Underneath this subtree, you can find the SQL logins. Here, you can change passwords or add new logins by right-clicking the Logins icon. Click this icon to open an option to **Add login**. After you open this option, type the

user name and password for the new login, as well as which databases the login is allowed to access. You must grant the test account access to the database you specify in the EAV configuration.

Running user-added EAVs

You may add your own monitors to those contained in **/user/local/lib/pingers.** For running these added programs, the monitor template **external** is used. The executable program is specified as the value of the attribute **run**. By default, the monitor looks for the run program in **/user/local/lib/pingers**. If the program resides elsewhere, a fully qualified path name must be entered. Any command line arguments to be used with the program are entered as **args** values. For example, suppose the program **my_pinger** is to be run with a **-q** option, so that it would be entered on the command line as follows:

```
my_pinger -q
```

This monitor might be specified as follows:

```
b monitor custom '{ use external run "my_pinger" args "-q" }'
```

Alternatively, you may pass arguments to the external monitor as environment variables. For example, you might want to enter this command:

```
/var/my_pinger /www/test_files/first_test
```

This could be specified in the conventional manner:

```
b monitor custom '{ use external run "/var/my_pinger" args "www/test_files/first_test" }'
```

It could also be specified in this way:

```
b monitor custom '{ use external run "/var/my_pinger" DIRECTORY "www/test_files" FILE
    "first_test" }'
```

This defines the monitor as shown in Figure 4.70.

```
monitor custom {
    use external
    run "/var/my_pinger"
    DIRECTORY "www/test_files"
    FILE "first_test" }
```

Figure 4.70 *Monitor template for an external monitor*

This frees the monitor definition from the rigidity of a strictly ordered command line entry. The arguments are now order-independent and may be used or ignored by the external executable.

Showing, disabling, and deleting monitors

You can show, disable, and delete monitors using the Configuration utility or from the command line. Deleting a monitor removes it from the **/config/bigip.conf** file. Disabling a monitor instance simply removes that instance from service until it is re-enabled. Disabling a monitor (which can be performed only at the command line) disables all instances of the monitor. All monitor instances are enabled by default.

To show or delete a monitor using the Configuration utility

1. In the navigation pane, click **Monitors**.
   A screen opens that lists monitors in two columns, System Supplied and User Defined.
2. To show a monitor, simply click the monitor name.
3. To delete a monitor, click the **Delete** button for the monitor. Note that only user-defined monitors can be deleted.

To show a monitor from the command line

You can display a selected monitor or all monitors using the **bigpipe monitor show** command:

```
b monitor <name> show
b monitor show all
```

To delete a monitor from the command line

You can delete a selected monitor using the **bigpipe monitor delete** command:

```
b monitor <name> delete
```

To disable a monitor instance using the Configuration utility

1. In the navigation pane, click **Monitors**.
   The Monitors screen opens.
2. Click the appropriate tab for the monitor instances: Basic Associations, Node Associations, or Node Address Associations.
   The resulting screen shows the existing associations (monitor instances).
3. Click the node you want to disable.
   The Properties screen for that node opens.
4. In the Monitor Instances portion of the screen, clear the **Enable** check box.
5. Click **Apply**.
   The monitor instance is now disabled.

To disable a monitor or monitor instance from the command line

To disable a monitor, use the **bigpipe monitor <name> disable** command:

```
b monitor <name> disable
```

This has the effect of disabling all instances of the monitor, as shown in Figure 4.71.

```
+- NODE  11.12.11.20:80   UP
|    |
|    +- http
|       11.12.11.20:80 up disabled
|
+- NODE  11.12.11.21:80   UP
|    |
|    +- http
|       11.12.11.21:80 up disabled
|
+- NODE  11.12.11.22:80   UP
     |
     +- http
        11.12.11.22:80 ip disabled
```

Figure 4.71 *All monitor instances disabled*

To disable a monitor instance, use the **bigpipe monitor instance <addr:port> disable** command:

```
b monitor instance <addr:port> disable
```

Disabled monitors and instances may be re-enabled as follows:

```
b monitor <name> enable
b monitor instance <addr:port> enable
```

To delete a monitor with no node associations from the command line

You can delete a monitor if it has no existing node associations and no references in a monitor rule. To delete a monitor, use the **bigpipe monitor <name> delete** command:

```
b  monitor my_http delete
```

If the monitor has instances, the instances must first be deleted using the **bigpipe node <addr:port> monitor delete** command. (Refer to *Showing and deleting associations*, on page 4-159.)

Associating the monitor with a node or nodes

Now that your monitor exists, the final step is to associate it with the nodes to be monitored. This creates an instance of the monitor for each node. At the command line, association is done using **bigpipe node** command:

```
b node <addr_list> monitor use <name>
```

For example, to associate monitor **http** with nodes **11.12.11.20:80, 11.12.11.21:80,** and **11.12.11.22:80**, the **bigpipe node** command would be as follows:

```
b node 11.12.11.20:80 11.12.11.21:80 11.12.11.22:80 monitor use http
```

This creates a monitor instance of **http** for each of these nodes. You can verify this association using the **bigpipe monitor show** command:

```
b node monitor show
```

This would produce the output shown in Figure 4.72.

```
+- NODE  11.12.11.20:80   UP
|    |
|    +- http
|       11.12.11.20:80 up enabled
|
+- NODE  11.12.11.21:80   UP
|    |
|    +- http
|       11.12.11.21:80 up enabled
|
+- NODE  11.12.11.22:80   UP
     |
     +- http
        11.12.11.22:80 ip enabled
```

Figure 4.72 *The output of the* ***b node monitor show*** *command*

The actual monitor instance for each node is represented by the output lines highlighted with bold text in Figure 4.72.

Reviewing types of association

While the term ***node association*** is applied generally, there are three types of association based on whether the monitor is associated with an address and a port, address(es) only, or port only. These are ***node association***, ***address association***, and ***port association***.

- Node association, strictly defined, is the association of a monitor with an address and port.
- Address association is the association of a monitor with an address only.
- Port association is the association of a monitor with a port only. For a port association, a wildcard character (*) is used to represent all addresses.

Once a monitor has been associated with a node, address, or port, no other monitor can be associated with the same node, address or port. However, an address association does not prevent a monitor from being associated with a node of the same address, or the reverse.

Using a simple association

The **http** example given above is the simplest kind of association, a node association performed using a monitor with a dest value of *:*. It can be seen in Figure 4.57, on page 4-141, that in each case the instance destination node is identical to the node the monitor has been associated with. This is because the template **http**, shown in Figure 4.57, was used as is, with a **dest**

value of ***:***. Either or both wildcard symbols can be replaced by an explicit **dest** value by creating a new monitor based on **http**. This is referred to as ***node*** and ***port aliasing***, described in the following section.

Using node and port aliasing

Usually the health of a node is checked by pinging that node. For this reason the **dest** attribute in the monitor template is always set to **"*"** or **"*:*"**. This causes the monitor instance created for a node to take that node's address or address and port as its destination. An explicit **dest** value forces the instance destination to a specific address and/or port which may not be that of the node. This causes the monitor to ping that forced destination by an unspecified path. Suppose, for example, that the association performed using **http** instead used a monitor **my_http** with a **dest** value of ***:443**. The node association command would be identical except that **http** is now replaced with **my_http**:

```
b node 11.12.11.20:80 11.12.11.21:80 11.12.11.22:80 monitor use my_http
```

This creates three instances of the monitor with the following **dest** values as shown in Figure 4.73.

```
+- NODE  11.12.11.20:80   UP
|    |
|    +- my_http
|       11.12.11.20:443 up enabled
|
+- NODE  11.12.11.21:80   UP
|    |
|    +- my_http
|       11.12.11.21:443 up enabled
|
+- NODE  11.12.11.22:80   UP
     |
     +- my_http
        11.12.11.22:443 up enabled
```

Figure 4.73 *Node ports aliased*

This is referred to as ***port aliasing***. The node itself can also be aliased, by assigning an explicit address to **dest**. For example, **dest** could set to **11.11.11.1:80**. This is called ***node aliasing*** and for the nodes

11.12.11.20:80, **11.12.11.21:80**, and **11.12.11.21:80** it would produce the following instances, (which are in fact one instance associated with three different nodes) as shown in Figure 4.74.

```
+- NODE  11.12.11.20:80   ADDR UP
|    |
|    +- my_http
|       11.11.11.1:80 checking enabled
|
+- NODE  11.12.11.21:80   ADDR UP
|    |
|    +- my_http
|       11.11.11.1:80 checking enabled
|
+- NODE  11.12.11.22:80   ADDR UP
     |
     +- my_http
        11.11.11.1:80 checking enabled
```

Figure 4.74 *Node addresses aliased*

Using transparent mode

Sometimes it is necessary to ping the aliased destination through a transparent node. Common examples are checking a router, or checking a mail or FTP server through a firewall. For example, you might want to check the router address **10.10.10.53:80** through a transparent firewall **10.10.10.101:80**. To do this, you would specify **10.10.10.53:80** as the monitor **dest** address (a node alias) and add the flag **transparent**:

```
b monitor http_trans '{ use http dest 10.10.10.53:80 transparent }'
```

Then you would associate the monitor **http_trans** with the transparent node:

```
b node 10.10.10.101:80 monitor use http_trans
```

This causes the address **10.10.10 53:80** to be checked through **10.10.10.101:80**. (In other words, the check of **10.10.10.53:80** is routed through **10.10.10.101:80**.) If the correct response is not received from **10.10.10.53:80**, then **10.10.10.101:80** is marked **down**.

◆ Note

Transparent mode applies only to the ECV monitors and to ***tcp_echo***.

Using logical grouping

In the preceding examples, only one monitor has been associated with the nodes. You may associate more than one monitor with a node or nodes by joining them with the Boolean operator **and**. This creates a rule, and the node is marked as **down** if the rule evaluates to false, that is, if not all the checks are successful. The most common example is the use of an HTTP monitor and an HTTPS monitor:

```
b node 11.12.11.20:80  monitor use my_http and my_https
```

The monitors themselves must be configured with the grouping in mind. For example, if the **dest** values of both monitors were set to ***:***, then both monitor instances would try to ping the default port **80**. This would both defeat the purpose of the HTTPS monitor and cause an automatic failure, since two monitors would be trying to ping the same address and port simultaneously.

Instead, monitor **my_http** should be given a **dest** value of ***:*** and monitor **my_https** should be given a **dest** value of ***:443**. This causes only the **my_http** monitor instances to default to **80**. The **my_https** monitor instances are forced to the explicit port **443**, avoiding a conflict as shown in Figure 4.75.

```
MONITOR  my_http and https_443
|
+- NODE  11.12.11.20:80   UP
|    |
|    +- my_http
|    |  11.12.11.20:80 up enabled
|    |
|    +- https_443
|       11.12.11.20:443 up enabled
```

Figure 4.75 *Use of a monitor rule*

Using wildcards to specify addresses

The wildcard * can be used to specify addresses. A wildcard address association creates instances of the monitor for all nodes load balanced by the BIG-IP:

```
b node * monitor use my_tcp_echo:
```

A wildcard with a port association creates instances of the monitor for every node configured to service that port:

```
b node *:80 monitor use my_http:
```

To associate a monitor using the Configuration utility

1. In the navigation pane, click **Monitors**.
   The Network Monitors screen opens.
2. Click one of three tabs:
   - If you are associating the monitor with a *node* (the IP address plus the port) click the Node Associations tab.
   - If you are associating the monitor with a *node address* only (the IP address minus the port), click the Node Address Associations tab.
   - If you are associating the monitor with a port only (the port minus the IP address), click the Port Associations tab.
3. Regardless of the selection you made in step 2, a dialog box appears with the boxes **Choose Monitor** and **Monitor Rule**. Type the monitor name or select one from the list.

4. If you want to associate more than one monitor, click the Move >> button to add the monitor name to the **Monitor Rule** box.
5. Repeat the previous two steps for each monitor you want to associate with a node.
6. Click **Apply** to associate the monitor(s).
   For additional information associating a monitor, click the **Help** button.

Showing and deleting associations

There are node commands for showing, and deleting node associations.

To show or delete associations using the Configuration utility

1. In the navigation pane, click **Monitors**.
   The Network Monitors screen opens.
2. Click one of three tabs:
   - If you are showing or deleting a *node* association (a node is the IP address plus the port) click the Node Associations tab.
   - If you are showing or deleting a *node address* association (the IP address minus the port), click the Node Address Associations tab.
   - If you are showing or deleting a *port* association (the port minus the IP address), click the Port Associations tab.
     Regardless of the selection you made in step 2, a dialog box opens showing existing associations, and with a **Delete Existing Associations** check box.
3. Delete associations by checking the box, then clicking **Apply**.

   Note: For wildcard address associations, the wildcard (*) association itself is shown in addition to each of the individual associations it produces. To delete all these associations, it is necessary to delete the wildcard association.

To show associations from the command line

You can display a selected node association or all node associations using the **bigpipe node monitor show** command:

```
b node monitor show
b node <addr:port> monitor show
```

To delete associations from the command line

You can delete a selected node association or all node associations using the **bigpipe node monitor delete** command:

```
b node <addr:port> monitor delete
```

In deleting specific monitor instances, it is important to consider how the association was made. If a monitor instance was created using a wildcard address, the wildcard must be deleted. For example, if multiple associations were created by entering **b node *:80 monitor use my_tcp_echo**, you would delete it by typing:

```
b node *:80 monitor delete
```

If multiple associations were created by entering **b node * monitor use my_tcp_echo**, you would delete it by typing:

```
b node * monitor delete
```

5

Configuring Filters

- Introduction
- IP filters
- Rate filters and rate classes

Introduction

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

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

Filter options are shown in Table 5.1.

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

Table 5.1 *The attributes you can configure for a filter*

Filtering should be kept to the minimum necessary, as filters may adversely affect performance.

Rate filters that limit traffic can have an adverse effect on monitors. If you have a large number of monitors configured, and the filters limit the monitor traffic, the monitor will mark the service as ***down****.*

IP filters

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

For a single filter, you can define multiple criteria in multiple, separate statements. Each of these statements should reference the same identifying name or number, to tie the statements to the same filter. You can have as many criteria statements as you want, limited only by the available memory. Of course, the more statements you have, the more difficult it is to understand and maintain your filters.

Configuring IP filters

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

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

To define an IP filter using the Configuration utility

1. In the navigation pane, click **Filters**.
   The IP Filters screen opens.
2. In the IP Filters screen, click the **Add** button.
   The Add IP Filter screen opens.
3. In the Add IP Filter screen, fill in the fields to define the filter. For additional information about defining an IP filter, click the **Help** button.

◆ Note

*For information on configuring IP filters from the command line, refer to the IPFW man page by typing **man ipfw** at the command prompt. You can configure more complex filtering by using the IPFW command line interface than you can from the Configuration utility.*

*Any **ipfw**-specific settings will be removed if you subsequently modify the filter using the Configuration utility.*

Rate filters and rate classes

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

Configuring rate filters and rate classes

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

◆ Tip

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

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

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

To configure a new rate class using the Configuration utility

1. In the navigation pane, click **Filters**.
   The IP Filters screen opens.
2. Click the Rate Filters tab.
   The Rate Filters screen opens.
3. Click the **Add Class** button.
   The Add Rate Class screen opens.
4. Type the necessary information to configure a new rate class. For additional information about configuring a new rate class, click the **Help** button.

◆ Note

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

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

To configure a rate filter using the Configuration utility

1. In the navigation pane, click **Filters**.
   The IP Filters screen opens.
2. Click the Rate Filters tab.
   The Rate Filters screen opens.
3. Click the **Add Filter** button.
   The Add Rate Filter screen opens.
4. Type the necessary information to configure a new rate filter. For additional information about configuring a rate filter, click the **Help** button.

◆ Note

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

6

Configuring a Redundant System

- Introduction
- Synchronizing configurations between units
- Configuring fail-safe settings
- Mirroring connection information
- Using gateway fail-safe
- Using network-based fail-over
- Setting a specific BIG-IP to be the preferred active unit
- Setting up active-active redundant BIG-IP units

Introduction

A BIG-IP redundant system consists of two identically configured BIG-IP units, only one of which is active at a given time (unless a special active-active configuration is chosen). The inactive unit serves as a standby which becomes active only in case of failure of the active system, a process called ***failover***.

BIG-IP redundant systems have special settings that you need to configure, such as VLAN fail-safe settings. One convenient aspect of configuring a redundant system is that once you have configured one of the BIG-IP units, you can simply copy the configuration to the other BIG-IP in the system by using the configuration synchronization feature.

There are two basic aspects about working with redundant systems:

- Synchronizing configurations between two BIG-IP units
- Configuring fail-safe settings for the VLANs

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

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

The attributes you can configure for a redundant systems are shown in Table 6.1.

Attributes	Description
Synchronizing configurations	This feature allows you to configure one BIG-IP and then synchronize the configuration with the other BIG-IP.
Fail-safe for VLANs	Fail-safe for VLANs provides the ability to cause a BIG-IP to fail over if a VLAN is no longer generating traffic.
Mirroring connections and persistence information	You can mirror connection and/or persistence information between redundant units. This enables you to provide seamless fail-over of client connections.
Gateway fail-safe	This feature allows you to fail-over between two gateway routers.
Network-based fail-over	You can configure the BIG-IP to use the network to determine the status of the active unit.

Table 6.1 *The attributes you can configure for redundant systems*

Attributes	Description
Setting a dominant BIG-IP	You can set up one unit in a pair to be the dominant active BIG-IP. The unit you set up as the dominant BIG-IP will always attempt to be active.
Active-active configuration	The default mode for a BIG-IP redundant system is Active/Standby. However, you can configure both units to run in active mode.

Table 6.1 *The attributes you can configure for redundant systems*

Synchronizing configurations between units

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

- The common **BIG/db** keys
- All files in **/config** (except **bigip_base.conf**)

If you use command line utilities to set configuration options, be sure to save the current configuration to the file before you use the configuration synchronization feature. (Alternately, if you want to test the memory version on the standby unit first, use **bigpipe config sync running**.)

Use the following **bigpipe** command to save the current configuration:

```
b save
```

◆ Note

*The BIG-IP software creates a file named **/usr/local/ucs/cs_backup.ucs** prior to installing a configuration file (UCS) from a remote machine.*

To synchronize the configuration using the Configuration utility

1. In the navigation pane, click **System**.
   The Network Map screen opens.
2. Click the Redundant Properties tab.
   The Redundant Properties screen opens.
3. Click the **Synchronize Configuration** button.

To synchronize the configuration from the command line

Synchronize the configuration from the command line using the **bigpipe config sync** command. Use the **bigpipe config sync** command without the **all** option to synchronize only the boot configuration file **/config/bigip.conf**.

The **bigpipe config sync all** command synchronizes the following configuration files:

- The common **BIG/db** keys
- All files in **/config** (except **bigip_base.conf**)

The **config sync running** command synchronizes the running version of **/config/bigip.conf**, which is the image that resides in memory as the system runs. This file is written only to memory on the standby unit, it is not saved.

Configuring fail-safe settings

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

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

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

Arming or disarming fail-safe on a VLAN

Each interface card installed on the BIG-IP is typically mapped to a different VLAN, which you need to know when you set the fail-safe option on a particular VLAN. You can view VLAN names in the Configuration utility, or you can use the **bigpipe vlan show** command to view VLAN names from the command line.

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

1. In the navigation pane, click **Network**.
   The VLANs list opens and displays all VLANs.
2. Select a VLAN name.
   The VLAN Properties screen opens.
3. To arm fail-safe, check **Arm Failsafe**.
4. To disarm fail-safe, clear the **Arm Failsafe** box.

5. If you are arming fail-safe, in the **Timeout** box, type the maximum time allowed for a loss of network traffic before a fail-over occurs.
6. Click the **Apply** button.

To arm or disarm fail-safe on a VLAN from the command line

To look up the names of the existing VLANs, use the **bigpipe vlan** command with the **show** keyword:

```
b vlan show
```

To arm fail-safe on a particular VLAN, use the **bigpipe vlan** command with the **failsafe arm** keyword:

```
b vlan <vlan_name> timeout <seconds>
b vlan <vlan_name> failsafe arm
```

For example, you have an external VLAN named **vlan1** and an internal VLAN named **vlan2**. To arm the fail-safe option on both cards with a timeout of 30 seconds, you need to issue the following commands:

```
b vlan vlan1 timeout 30
b vlan vlan2 timeout 30
b vlan vlan1 failsafe arm
b vlan vlan2 failsafe arm
```

To disarm fail-safe on a particular VLAN, use the **bigpipe vlan** command with the **failsafe arm** keyword:

```
b vlan <vlan_name> failsafe disarm
```

Mirroring connection information

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

The ***mirror*** feature on BIG-IP units is the ongoing communication between the active and standby units that duplicates the active unit's real-time connection information state on the standby unit. If you have enabled mirroring, fail-over can be so seamless that file transfers can proceed uninterrupted, customers making orders can complete transactions without interruption, and your servers can generally continue with whatever they were doing at the time of fail-over.

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

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

Commands for mirroring

Table 6.2 contains the commands that support mirroring capabilities. For complete descriptions, syntax, and usage examples, see Chapter 7, *bigpipe Command Reference*.

bigpipe command	Options
b global mirror	Options for global mirroring
b virtual mirror	Options for mirroring connection and persistence information on a virtual server.
b snat mirror	Options for mirroring secure NAT connections

Table 6.2 *Mirroring commands in* ***bigpipe***

To configure global mirroring

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

```
b global mirror enable | disable | show
```

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

```
b global mirror enable
```

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

```
b global mirror disable
```

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

```
b global mirror show
```

Mirroring virtual server state

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

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

◆ Note

Mirroring cannot be used with SSL gateways

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

```
b virtual <virt addr>:<service> \
mirror [conn] enable|disable
```

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

```
b virtual <virt addr>:<service> \
mirror [conn] show
```

If you do not specify **conn** for connection information, the BIG-IP assumes that you want to display this type of information.

Mirroring SNAT connections

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

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

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

```
b snat 192.168.225.100 mirror enable
```

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

```
b snat default mirror enable|disable
```

Using gateway fail-safe

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

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

Adding a gateway fail-safe check

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

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

◆ Note

We recommend a gateway failsafe ping interval of 2 seconds with a timeout of 10 seconds. If this interval is too small, you can use any 1 to 5 ratio that works for you.

To configure gateway fail-safe using the Configuration utility

1. In the navigation pane, click **System**.
   The Network map screen opens.
2. Click the Redundant Properties tab.
   The Redundant Properties screen opens.
3. In the Gateway Fail-safe section of the screen, make the following entries:
   - Check the **Enabled** box.
   - In the **Router** box, type the IP address of the router you want to ping.
   - In the **Ping (seconds)** box, type the number of seconds you want the BIG-IP to wait before it pings the router.
   - In the **Timeout (seconds)** box, type the timeout value, in seconds. If the router does not respond to the ping within the number of seconds specified, the gateway is marked **down**.
4. Click the **Apply** button.

To configure gateway fail-safe in BIG/db

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

- The IP address of the router
- The ping interval
- The timeout period

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

```
b db set Local.Bigip.GatewayPinger.Ipaddr=<gateway IP>
```

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

```
b db set Local.Bigip.GatewayPinger.Pinginterval=<seconds>
```

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

```
b db set Local.Bigip.GatewayPinger.Timeout=<seconds>
```

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

```
bigstart reinit bigd
```

For more information about BIG/db and using **bigpipe db**, see Chapter 9, *BIG/db Configuration Keys*.

To enable gateway fail-safe from the command line

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

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

```
b global gateway failsafe arm
```

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

```
b global gateway failsafe disarm
```

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

```
b global gateway failsafe show
```

Finding gateway fail-safe messages

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

Using network-based fail-over

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

To configure network-based fail-over using the Configuration utility

1. In the navigation pane, click **System**.
   The Network Map screen opens.
2. Click the Redundant Properties tab.
   The Redundant Properties screen opens.
3. Check the **Network Failover Enabled** box.
4. Click the **Apply** button.

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

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

```
b db set Common.Sys.Failover.Network=1
b failover init
```

Other keys are available to lengthen the delay to detect the fail-over condition on the standby unit, and to lengthen the heart beat interval from the active unit. The default number of seconds required for the standby unit to notice a failure in the active unit is **3** seconds. To change the default setting, use the following syntax:

```
b db set Common.Bigip.Cluster.StandbyTimeoutSec=<value>
b failover init
```

The default heart beat interval is **1** second. To change it from the active BIG-IP, change the following value using **b db:**

```
b db set Common.Bigip.Cluster.ActiveKeepAliveSec=<value>
b failover init
```

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

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

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

The active or standby preference for the BIG-IP is defined by setting the appropriate startup parameters for the fail-over mechanism in BIG/db. For more details on fail-over startup and functioning, see *Failover and cluster keys*, on page 9-2.

To force a BIG-IP to active or standby state

The following example shows how to set the BIG-IP to standby:

```
b db set Local.Bigip.Failover.ForceStandby
b failover init
```

A BIG-IP that prefers to be standby can still become the active unit if it does not detect an active unit.

This example shows how to set a BIG-IP to active:

```
b db set Local.Bigip.Failover.ForceActive
b failover init
```

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

Setting up active-active redundant BIG-IP units

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

◆ Note

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

MAC masquerading is not supported in active-active mode.

Configuring an active-active system

The default mode for BIG-IP redundant systems is active/standby. To use active-active mode on the BIG-IP redundant system, you must perform the following tasks, in order. Each task included below is outlined in the following sections.

- Enable active-active on the BIG-IP.
- Configure an additional floating self IP address on the internal VLAN for each unit. You must have two floating self IP addresses for the redundant system.
- Set the routing configuration on the servers that are load balanced by the active-active BIG-IP system.
- Make sure the BIG/db key **Local.Bigip.Failover.UnitId** is set at **1** for one of the units, and **2** for the other.
- Define the virtual servers, NATs, and/or SNATs to run on either unit **2** or **1**.
- Update the fail-over mechanism with the configuration changes made in BIG/db.
- Synchronize the configuration.
- Complete the transition from active/standby to active-active.

Task 1: Enabling active-active on the BIG-IP

The first task you need to complete is to enable active-active on the BIG-IP in the redundant system.

To enable active-active using the Configuration utility

Perform this procedure on the active unit first. After the active unit is enabled, wait several seconds and open the Configuration utility for the other unit. Follow this procedure on the other unit. After you perform this task on the standby unit, wait several seconds and click the **Refresh** button (Microsoft Internet Explorer) or **Reload** button (Netscape Navigator) on the browser for both units.

1. In the navigation pane, click **System**.
   The Network Map screen opens.
2. Click the Redundant Properties tab.
   The Redundant Properties screen opens.
3. Check the **Active-Active Mode Enabled** box.
4. Click the **Apply** button.

To enable active-active from the command line

Set the **Common.Bigip.Failover.ActiveMode** key to **1**. Use the following commands on each unit to enable active-active mode:

```
b db set Common.Bigip.Failover.ActiveMode = 1
b failover init
```

The default for this entry is **0** which indicates that the unit is in active/standby mode.

Task 2: Configuring an additional floating self IP address

When you configure a redundant system, you enter a pair of shared floating self IP addresses, one for the external VLAN and one for the internal VLAN. As defined during Setup utility configuration, the floating self IP addresses are configured as belonging to unit one and unit two.

In an active-active configuration, each BIG-IP must have its own floating self IP address on the internal VLAN. This is the address to which the servers behind the BIG-IP route traffic. For example, you could use **11.12.11.3** as the internal floating self IP address for unit one and **11.12.11.4** as the internal floating self IP address for unit two. Configured correctly, **11.12.11.3** should appear on both units as a floating self IP address belonging to unit one and **11.12.11.4** should appear on both units as a floating self IP address belonging to unit two.

Since you already have a floating self IP address for the internal interface that is configured for unit one and unit two, you only need to create a second floating IP address for unit two.

To create an additional floating self IP address

On unit two, create the second internal floating self IP address and assign it to unit two:

```
b self 11.12.11.4 vlan internal unit 2 floating enable
```

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

You can configure additional shared IP aliases on the external VLANs of each BIG-IP, as well. This makes it possible for routers to route to a virtual server using virtual **noarp** mode.

Task 3: Configuring servers for active-active

In an active-active system, the servers must be logically segregated to accept connections from one BIG-IP or the other. To do this, set the default route of some of your servers to the floating self IP address on one unit and the default route on some of your servers to the floating self IP address on the other unit (see *Task 2: Configuring an additional floating self IP address*). When a unit fails over, the surviving BIG-IP assumes the internal IP alias of the failed machine, providing each server a default route.

Task 4: Checking the BIG-IP unit number

Using the **bigpipe db get *unit*** command, check the value of the BIG/db key **Local.Bigip.Failover.UnitId**. This value should be **1** for one of the units, and **2** for the other.

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

◆ Note

You set the unit number for the BIG-IP in the Setup utility.

To check the BIG-IP unit number using the Configuration utility

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

1. Open the Configuration utility.
2. In the navigation pane, look at the upper left corner. The status of the BIG-IP is **Active** and the unit number is either **1** or **2**.

Task 5: Defining the virtual address configuration

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

To define virtual servers, NATs, and SNATs on active-active units from the command line

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

```
b virtual <virt addr>:<service> [unit <1|2>] \
use pool <pool_name>|rule <rule_name>
b nat <internal_ip> to <external_ip> ... [unit <1|2>]
b snat map <orig_ip> to <snat_ip> ... [unit <1|2>]
```

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

Note

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

Note

The default SNAT is not compatible with an active-active system. However, you may create the equivalent of a default SNAT using SNAT automap. Refer to ***Enabling or disabling SNAT automap****, on page 3-19.*

To define virtual servers, NATs, and SNATs on active-active units using the Configuration utility

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

1. In the navigation pane, click **Virtual Servers**.
   The Virtual Servers screen opens.
2. Click the **Add** button.
3. When you reach the Configure Redundant Properties screen, select the unit number for the virtual server from the **Unit ID** list. The connections served by this virtual server are managed by the BIG-IP assigned to this unit ID.
4. Complete the Resources section of the screen. For more information about individual settings, refer to the online help.
5. Click the **Apply** button.

Task 6: Updating the fail-over mechanism with the configuration changes made in BIG/db

If you change a BIG/db key that affects the fail-over mechanism (keys that contain the word Failover) the system needs to be updated with the change. To update the fail-over mechanism, type the following command:

```
b failover init
```

Task 7: Synchronizing the configuration

After you complete all six previous tasks on each BIG-IP in the active-active system, synchronize the configurations on the units with the Configuration utility, or from the command line.

To synchronize the configuration using the Configuration utility

1. In the navigation pane, click **System**.
   The Network Map screen opens.
2. Click the **Redundant Properties** tab.
   The Redundant Properties screen opens.
3. Click the **Synchronize Configuration** button.

To synchronize the configuration from the command line

To synchronize the configuration between two BIG-IP units from the command line, use the following command:

```
b config sync all
```

Task 8: Transitioning from active/standby to active-active

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

```
b failover standby
```

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

```
b failover standby
```

Now both units are in active-active mode.

◆ Note

This task is not required if you enable active-active in the Configuration utility. The transition is made during ***Task 1: Enabling active-active on the BIG-IP****, on page 6-11.*

Understanding active-active system fail-over

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

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

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

Disabling automatic fail back

In some cases, you may not want connections to automatically failback. The fact that a machine has resumed operation may not be reason enough to disrupt connections that are running on the BIG-IP serving as both units. Note that because of addressing issues, it is not possible to slowly drain away connections from the machine that was running as both units, giving new requests to the recently rebooted machine.

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

Taking an active-active BIG-IP out of service

You can use the bigpipe **failover standby** command to place an active unit in standby mode. In active-active mode, type the following command to place one of the active units in standby mode:

```
b failover standby
```

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

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

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

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

```
b failover failback
```

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

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

Introducing additional active-active BIG/db configuration parameters

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

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

- **Local.Bigip.Failover.UnitId**
  This is the default unit number of the BIG-IP. This value is set by the Setup utility or when you upgrade your units to this version of the BIG-IP software.
- **Common.Bigip.Failover.ManFailBack**
  This is set to **1** so that manual intervention is required (the **bigpipe failover failback** command is issued) before a BIG-IP running both unit numbers surrenders a unit number to its peer. This feature is **off** by default, fail-back is automatic. For more details, see the section *Understanding active-active system fail-over*, on page 6-15.
- **Common.Bigip.Failover.AwaitPeerDeadDelay**
  The BIG-IP checks to see that its peer is still alive at this rate (in seconds). The default value for this parameter is one second.
- **Common.Bigip.Failover.AwaitPeerAliveDelay**
  Check status of a peer BIG-IP while waiting for it to come to life with this frequency (in seconds). The default value of this parameter is three seconds.
- **Common.Bigip.Failover.DbgFile**
  If a file name is specified, the fail-over mechanism logs state change information in this file. This value is not set by default.
- **Common.Bigip.Failover.PrintPeerState**
  Causes the fail-over mechanism to periodically write to the log file, the state of its connections to its peer (hard-wired and/or network). **Common.Bigip.Failover.DbgFile**.

Additional commands for displaying active vs. mirrored data

The **dump** commands explicitly show those connections (and other objects) that are active on the BIG-IP, and those that are standby connections for the peer BIG-IP. In prior versions of the BIG-IP, one unit is the active unit and the other is the standby. When the **bigpipe conn dump** command is issued on the active unit, each of the connections shown is active. Similarly, when the **bigpipe conn dump** command is issued on the standby unit, it is clear that each of the connections listed is a standby connection. These standby connections are created by mirroring the active connections on the standby unit.

In an active-active installation, each unit can be considered a standby for its peer BIG-IP. By default, the **dump** command only shows items that are active on the given unit. To see standby items you must use the **mirror** qualifier. You can use the following commands with the mirror option:

```
b conn dump [mirror]
b virtual persist dump [mirror]
b sticky dump [mirror]
```

Also, we have modified the **bigpipe snat show** command output to show whether a connection listed is an active connection or a mirror connection.

Reviewing specific active-active bigpipe commands

There are several specific commands included in **bigpipe** to support active-active configurations. One of these commands is the **bigpipe failover init** command. You can use the **bigpipe failover init** command to read the BIG/db database and refresh its parameters. To do this, type the following command:

```
b failover init
```

Another command specifically designed for active-active configurations is the **bigpipe failover failback** command. This command causes the BIG-IP to resume normal operation after a fail-over occurs. To do this, type the following command:

```
b failover failback
```

After a **bigpipe failover standby** command is issued, use this command to allow the BIG-IP to resume normal operation. If manual fail back is enabled, this command causes a BIG-IP that is running as both units to release a unit number to its peer unit when the peer becomes active. You can use the following commands to view the unit number on the BIG-IP you are logged into:

```
b unit [show]
```

To view the unit number, or numbers, of the peer BIG-IP units in a redundant system, type the following command:

```
b unit peer [show]
```

Returning an active-active installation to active/standby mode

Returning to active/standby mode from active-active mode is relatively simple in that only a few things need be undone.

1. Enable active/standby mode by setting the BIG/db key **Common.Bigip.Failover.ActiveMode** to **0**.
2. Update the fail-over mechanism with the change by typing the command **bigpipe failover init**.
3. Synchronize the configuration by typing the command **bigpipe configsync all**.
4. Since each BIG-IP is an active unit, transition each unit into active/standby mode by typing the command **bigpipe failover standby** on each unit.

When in active/standby mode, the active BIG-IP runs all objects (virtual servers, SNATs and NATs) that are defined to run on unit 1 or unit 2. It is not necessary to redefine virtual servers, SNATS, or NATs when you transition from active-active mode to active/standby mode.

7

bigpipe Command Reference

bigpipe commands

This chapter lists the various **bigpipe** commands, including syntax requirements and functional descriptions. Table 7.1 outlines the conventions used in the command line syntax.

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

Table 7.1 *Command line conventions*

The following table provides a concise listing of the individual **bigpipe** commands, along with the page reference where you can find the detailed description.

Command	Description	Page
-?	Displays online help for an individual **bigpipe** command.	7-3
class	Displays all classes included with BIG-IP.	7-4
config	Synchronizes the **/config/bigip.conf** between the two BIG-IP units in a redundant system.	7-5
conn	Shows information about current connections such as the source IP address, virtual server and port, and node.	7-7
default_gateway	Creates a pool of default gateways.	7-8
failover	Sets the BIG-IP as active or standby.	7-9
global	Sets global variable definitions.	7-10
-h and help	Displays online help for **bigpipe** command syntax.	7-17
interface	Sets options on individual interfaces.	7-18
load	Loads the BIG-IP configuration and resets.	7-19
maint	Toggles the BIG-IP into and out of maintenance mode.	7-20

-?

```
b <command> -?
```

For certain commands, displays online help, including complete syntax, description, and other related information. For example, to see online help for the **bigpipe service** command, type:

```
b service -?
```

class

```
b class <class name> { <ip member> <ip member> ... } <ip member> ::= HOST <ip addr> |
    NETWORK <ip addr> MASK <ip addr>
b class <class name> { <string> <string> ... }
b class <class name> { <num> <num> ... }
b <class name> show
b class ip show
b class string show
b class value show
b class show
b class <class name> delete
```

Creates, shows, and deletes any classes, such as class AOL. Default classes are also shown.

The BIG-IP includes a number of predefined lists. They are:

- AOL Network
- Image Extensions
- Non-routable addresses

These lists are located in the file **/etc/default_classes.txt**. When the **bigpipe load** command is issued, the lists are loaded. Unless modified by a user, these lists are not saved to the file **bigip.conf**.

The following are examples of class types defined with the **class** command. Note that string classes require escape characters in the syntax to keep from being interpreted literally by the UNIX system.

```
b class string_class { \".string\" ... } | '{ ".string" }'
b class numeric_class { <num> <num> ... }
b class host_class { host <ip_addr> }
b class network_class { network <ip_addr> mask <mask_addr> }
```

config

```
b config sync
b config sync all
b config sync running
b config save <file>
b config install <file>
```

Synchronizes configurations of two BIG-IP units in a redundant system by collecting and copying the configuration file(s) from the active unit to the standby unit (**config sync**). Also archives configuration files for backup purposes (**config save**) and installs saved files (**config install**).

Synchronizing configuration files

config sync without the **all** option synchronizes only the basic configuration file **/config/bigip.conf**.

config sync all synchronizes the following configuration files:

- The common **BIG/db** keys
- All common files in **/config**
- All common files in **/etc**

config sync running synchronizes the running version of **/config/bigip.conf**, which is the image that resides in memory as the system runs. This file is loaded into memory on the standby unit, it is not saved.

◆ Note

*The **config sync** command applies only to BIG-IP and not to 3-DNS.*

Saving configuration files to an archive

config save <file> saves all configuration files to a single archive file, **<file>.ucs**, on the local unit without copying it to the standby unit. By default, **<file>.ucs** is saved to the directory **/user/local/ucs.** An alternate location can be specified by expressing **<file>** as a relative or absolute path. For example:

```
b config save /user/local/config_backup/my_conf
```

This writes the file **my_conf.ucs** to the directory **/user/local/config_backup**.

Installing an archived configuration file

config install <file> reinstalls the archived configuration files saved as **<file>.ucs** to their working locations on the local unit.

If you use command line utilities to set configuration options, be sure to save the current configuration to the relevant files before you use the configuration synchronization feature. (Alternatively, if you want to test the memory version on the standby unit first, use **bigpipe config sync running**.) Use the following **bigpipe** command to save the current configuration:

```
b save
```

Note

*A file named **/usr/local/ucs/cs_backup.ucs** is created prior to installing a UCS from a remote machine.*

conn

```
b conn [ <client_ip>[:<client_service>] ] dump [mirror]
```

Displays information about current client connections to virtual addresses and virtual servers.

The following command displays all current client connections:

```
b conn dump
```

The output shows the source IP address, virtual server IP address, and node to which the client is connected.

```
bigip conn dump

fromvirtualnode
100.100.100.30:49152 ->100.100.100.100:23 ->200.200.200.10:23
100.100.101.90:49153 ->100.100.100.100:80 ->200.200.200.10:80
...
```

Figure 7.1 *Formatted output of the* ***conn*** *command*

This command can also show connections that are active on the given BIG-IP, as well as those that are standby connections for the peer BIG-IP. By default, the **dump** command only shows items that are active on the given unit. To see standby items, you must use the **mirror** qualifier.

```
b conn dump mirror
```

default_gateway

```
b default_gateway  use pool <pool_name>
b default_gateway show
b default_gateway delete
```

This command creates, shows, or deletes a pool of default gateways, with nodes in the pool corresponding to different routes. Connections originating from the system with a destination for which there is no other route choose a route from the default gateway pool. Note that the default gateway pool is not a last-hop pool for services running on the system.

There can be only one default gateway pool at any one time.

Defining a default gateway pool removes the need to define a default route. However, if a default route is defined, that route will be used when all the nodes in the default gateway pool are down.

Since the system performs route lookups on nodes as they are defined, the default gateway pool must be stored at the top of the **bigip.conf** file. Also, all nodes in the default gateway pool must reside on the same IP network as the system.

We recommend that all nodes in the default gateway pool have the same MTU.

As an alternative to using the **default_gateway** command, you can use the Setup utility, which allows you to create the default gateway pool at the time that you configure your base network.

failover

```
b failover active | standby | show | init | failback
```

This group of commands affects the fail-over status of the BIG-IP or 3-DNS system.

In an active/standby or active-active configuration, run the following command to place a BIG-IP or 3-DNS system in standby mode:

```
b failover standby
```

Show the status of the BIG-IP or 3-DNS system with the following command:

```
b failover show
```

In an active-active configuration, run the following command after you issue the **bigpipe failover standby** command. This allows the inactive unit to resume handling connections:

```
b failover failback
```

◆ Note

*The **failback** command is only applicable if you are running a redundant system in active-active mode.*

You can use the **bigpipe failover init** command to refresh the parameters of the fail-over mechanism with any new configuration data entered in the BIG/db database.

```
b failover init
```

global

```
b global auto_lasthop enable | disable | show
b global fastest_max_idle_time <seconds>
b global fastflow_active auto | on | off | show
b global fastflow_active auto | on | off | show
b global gateway failsafe arm | disarm | show
b global ipforwarding enable | disable
b global mirror enable | disable | show
b global memory_reboot_percent <percent>
b global open_3dns_ports enable | disable | show
b global open_corba_ports enable | disable | show
b global open_snmp_ports enable | disable | show
b global open_telnet_port enable | disable
b global open_ftp_ports enable | disable
b global open_ssh_port enable | disable
b global open_rsh_ports enable | disable
b global open_failover_ports enable | disable | show
b global persist_map_proxies enable | disable
b global persist timer limit | timeout | show
b global persist across_services enable | disable
b global persist across_virtuals enable | disable
b global self_conn_timeout enable | disable | show
b global sslproxy serverssl cache timeout <num>
b global sslproxy serverssl cache size <num>
b global sslproxy serverssl failover <enable | disable>
b global sslproxy serverssl unclean shutdown <enable | disable>
b global sslproxy serverssl strict resume <enable | disable>
b global sticky table_limit <max_num> | show
b global verbose_log_level <level>
b global webadmin_port <port>
b global l2_aging_time <seconds>
```

auto_lasthop

When this variable is enabled, it automatically designates the lasthop router inside IP address as a lasthop route for replies to inbound traffic. If **auto_lasthop** is disabled, the lasthop router inside IP address must be specified as a **lasthop pool**. The default setting is **enable**.

fastest_max_idle_time

Sets the number of seconds a node can be left idle by the **fastest** load balancing mode. This forces the BIG-IP to send fewer connections to a node that is responding slowly, and also allows the BIG-IP to periodically recalculate the response time of the slow node.

fastflow_active

You can use this variable to control additional enhancements that speed packet flow for TCP connections when the packets are not fragmented. In most configurations these software enhancements are automatically turned on and do not require any additional configuration.

However, you may want to turn off these enhancements for individual virtual servers that use IPFW rate filters. With the speed enhancements **on**, IPFW only examines the first SYN packet in any given connection. If you want to filter all packets, you should turn the speed enhancements **off**. To do this, you first set the global state of the system **on**, and then you turn the feature **off** for individual virtual servers that use IPFW rate filtering. You can also change the settings for these enhancements from the command line or in the Configuration utility.

There are three global states you can set with **fastflow_active**. The default state is **auto**. The global states are:

- **off**
- **auto**
- **on**

The additional speed enhancements are globally disabled if the **sysctl** variable **fastflow_active** is **off** or if **fastflow_active** is set to **auto** and an IPFW rate filter exists in the configuration.

To provide the benefits of software acceleration for virtual servers that do not use rate filtering and turn off software acceleration for virtual servers that use IPFW rate filtering, you can set the global variable **fastflow_active** to **on** with the following command:

```
b global fastflow_active on
```

After you set the **sysctl** variable, use the following **bigpipe** command to disable software acceleration for virtual servers that use IPFW rate filtering:

```
b virtual <ip>:<port> accelerate disable
```

gateway failsafe

Turns the gateway fail-safe feature on and off. This command is supported only for redundant systems.

The typical use of gateway fail-safe is a setup where active and standby BIG-IP units use different routers as gateways to the Internet. Fail-over is triggered if the gateway for the active unit is unreachable.

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

```
b global gateway failsafe arm
```

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

```
b global gateway failsafe disarm
```

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

```
b global gateway failsafe show
```

For more information about configuring gateway fail-safe, see *Health monitors*, on page 4-137.

ip forwarding

Enables IP forwarding for the BIG-IP. IP forwarding exposes all of the node IP addresses to the external network, making them routable on that network. The default setting is **disabled**.

mirror

Enables mirroring functions globally for the BIG-IP. The mirror feature duplicates the active unit's real-time connection or persistence information state on the standby unit for smooth transition to the inactive unit at fail-over. The default setting is **enabled**.

memory_reboot_percent

The value you type, **80** or higher, is the percentage of memory that is in use before the BIG-IP automatically reboots. The default value for this variable is **95**. To disable this feature, set the value to **0**.

open_3dns_ports

This variable is required only when running one or more separate 3-DNS Controllers in the network. It does not apply to running the 3-DNS software module on the BIG-IP itself. The variable is disabled on the BIG-IP when the 3-DNS Controller is not present in the network configuration. (See the ***3-DNS Administrator Guide*** for more information.)

open_corba_ports

This variable enables and disables the CORBA ports, which allow administrative CORBA connections. The default setting is **disabled**.

open_snmp_ports

This variable enables and disables the SNMP ports, which allow administrative SNMP connections. The default setting is **disabled**.

open_telnet_port

This variable enables or disables ports for Telnet access, and the default setting is **disable**.

The following command sets this variable to open the Telnet port (**23**) to allow administrative Telnet connections. This is useful for BIG-IP units that do not support encrypted communications, or for a unit that needs to communicate with the 3-DNS software. (See the ***3-DNS Administrator Guide*** for more information.)

The following command opens the Telnet port:

```
b global open_telnet_port enable
```

The following command closes the Telnet port:

```
b global open_telnet_port disable
```

open_ftp_ports

This variable enables or disables ports for FTP access, and the default setting is **disable**.

The following command open the FTP ports (**20** and **21**) to allow administrative FTP connections, which is useful for BIG-IP units that do not support encrypted communications.

```
b global open_ftp_ports enable
```

The following command closes FTP ports:

```
b global open_ftp_ports disable
```

open_ssh_ports

This variable enables or disables ports for SSH access on BIG-IP units that support encrypted communication. The default setting is **enable**.

The following command opens the SSH port (**22**) to allow encrypted administrative connections:

```
b global open_ssh_port enable
```

The following command closes the SSH port:

```
b global open_ssh_port disable
```

open_rsh_ports

This variable enables or disables ports for RSH access, and it is useful for BIG-IP units that do not support encrypted communications, or for connecting to 3-DNS Controllers that do not support encrypted communication. (See the ***3-DNS Administrator Guide*** for more information.)

The default setting is **disable**.

The following command opens the RSH ports (**512**, **513**, and **514**) to allow RSH connections:

```
b global open_rsh_ports enable
```

The following command closes RSH ports:

```
b global open_rsh_ports disable
```

open_failover_ports

This variable enables or disables network failover when a VLAN has port lockdown enabled.

The following command enables network failover:

```
b global open_failover_ports enable
```

The following command disables network failover:

```
b global open_failover_ports disable
```

persist map_proxies

The default setting for the map proxies for the persistence variable is **enable**. The AOL proxy addresses are hard-coded. This enables you to use client IP address persistence with a simple persist mask, but forces all AOL clients to persist to the same server. All AOL clients will persist to the node that was picked for the first AOL client connection received.

The class B networks, **195.93** and **205.188**, are mapped to **152.163** for persistence. For example, client **195.93.3.4** would map to **152.63.3.4** for persistence records only. This mapping is done prior to applying the persist mask. Use **bigpipe pool persist dump** to verify that the mapping is working.

We recommend that in addition to setting this **sysctl** variable, you set a persist mask of **255.255.0.0** so that all the AOL addresses map to a common address. For example, Table 7.2 is an example of how setting this variable and a persist mask of **255.255.0.0** would map a sample set of client addresses.

Sample Client Address	Persist Address
152.44.12.3	195.93.0.0
152.2.99.7	195.93.0.0
170.11.19.22	195.93.0.0
202.67.34.11	195.93.0.0
205.188.11.2	195.93.0.0
208.33.23.4	208.33.0.0 (non AOL address is not mapped)

Table 7.2 *Address mapping of sample clients*

persist timer

The following command forces the persistent connection timer to reset on each packet for persistent sessions. This is the default value.

```
b global persist timer limit
```

The following command resets the timer only when the persistent connection is **initiated**.

```
b global persist timer timeout
```

◆ Note

For SSL persistence, the timer is always reset on each packet.

persist across_services

When this variable is enabled, all simple persistence connections from a client IP address that go to the same virtual address also go to the same node (matches the client address and the virtual IP address but not the virtual port).

The default setting for this variable is **disabled**.

persist across_virtuals

When this variable is enabled, all simple persistent connections from the same client IP address are sent to the same node (matches the client IP address but not the virtual address or virtual port the client is using). The default setting for this variable is **disabled**.

self_conn_timeout

This variable is used as a tracking mechanism for UDP connections. After the number of seconds specified by this variable has expired, the UDP connection terminates. The default value for this variable is **5**.

sticky table_limit

This is the maximum number of sticky entries allowed to accumulate on the BIG-IP when using destination address affinity (sticky persistence). When the maximum value is reached, the BIG-IP stops accumulating sticky entries. The default value for this entry is **2048**.

verbose_log_level

This variable sets logging levels for both TCP and UDP traffic. Each log level is identified by a level number used in place of the **<level>** parameter.

The following command turns on port denial logging for both TCP and UDP traffic. This logs TCP and UDP port denials to the virtual server address and the BIG-IP address.

```
b global verbose_log_level 15
```

The following command turns logging off altogether:

```
b global verbose_log_level 0
```

Setting log levels only for TCP traffic

The following command turns on only TCP port denial logging, which logs TCP port denials to the BIG-IP address.

```
b global verbose_log_level 2
```

The following command turns on virtual TCP port denial logging, which logs TCP port denials to the virtual server address.

```
b global verbose_log_level 8
```

Setting log levels for UDP traffic

The following command turns on only UDP port denial logging, which logs UDP port denials to the BIG-IP address.

```
b global verbose_log_level 1
```

The following command turns on only virtual UDP port denial logging, which logs UDP port denials to the virtual server address.

```
b global verbose_log_level 4
```

webadmin_port

Specifies the port number used for administrative web access. The default port for web administration is port **443**.

l2_aging_time

Specifies a time period after which dynamic entries in the L2 forwarding table are flushed out if the MAC address is no longer present on the network. The default value is **300** seconds.

-h and -help

```
b [-h | -help ]
```

Displays the **bigpipe** command syntax or usage text for all current commands.

◆ Note

*More detailed man pages are available for some individual **bigpipe** commands. To display detailed online help for the **bigpipe** command, type: **man bigpipe**.*

interface

```
b interface <if_name> media <media_type> | show
b interface <if_name> duplex full | half | auto | show
b interface [<if_name>] show [verbose]
b interface [<if_name>] stats reset
```

Displays names of installed network interface cards and allows you to set properties for each network interface card.

Setting the media type

The media type may be set to the specific media type for the interface card or it may be set to **auto** for auto detection. If the media type is set is set to **auto** and the card does not support auto detection, the default type for that interface will be used, for example **1000BaseTX**.

Setting the duplex mode

Duplex mode may be set to **full** or **half duplex**. If the media type does not allow duplex mode to be set, this will be indicated by an onscreen message. If media type is set to **auto**, or if setting duplex mode is not supported, the duplex setting will not be saved to the **bigip.conf** file.

load

```
b [verify] load [ <filename> | - ]
b [-log] load [ <filename> | - ]
```

Resets all of the BIG-IP settings and then loads the configuration settings, by default from the **/config/bigip.conf** and **/config/bigip_base.conf** files.

For testing purposes, you can save a test configuration by renaming it to avoid confusion with the boot configuration file. To load a test configuration, use the **load** command with the <**filename**> parameter. For example, if you renamed your configuration file to **/config/bigtest.conf**, the command would be:

```
b load /config/bigtest.conf
```

The command checks the syntax and logic, reporting any errors that would be encountered if the command executed.

You can type **b load -** in place of a file name, to display the configuration on the standard output device.

```
b save -
```

Use the **load** command together with the **verify** command to validate the specified configuration file. For example, to check the syntax of the configuration file **/config/altbigpipe.conf**, use the following command:

```
b verify load /config/altbigip.conf
```

The **-log** option will cause any error messages to be written to **/var/log/bigip** in addition to the terminal.

maint

```
b maint
```

Toggles a BIG-IP into and out of Maintenance mode. When in Maintenance mode, a BIG-IP accepts no new connections, but it does allow existing connections to complete.

The **maint** command interactively prompts you to enter or exit the maintenance mode.

```
b maint
```

If the BIG-IP is already in maintenance mode, the **maint** command takes the BIG-IP out of maintenance mode. If the BIG-IP is in maintenance mode for more than 20 minutes, that BIG-IP immediately begins to accept new connection requests.

If the BIG-IP has been in maintenance mode for more than 20 minutes, it automatically updates all network ARP caches; this process normally takes a few seconds. However, you can speed the process up by reloading the configuration file, using the following command:

```
b -f /config/bigip.conf
```

makecookie

```
b makecookie <ip_addr:service>
```

Generates a cookie string with encoding automatically added for cookie persistence Passive mode:

```
b makecookie <server_address:service> [ > <file>]
```

This command prints a cookie template similar to the templates shown in Figure 7.2 and Figure 7.3.

```
Set-Cookie:BIGipServer[poolname]=336268299.20480.0000; path=/
```

Figure 7.2 *Sample cookie template*

```
Set-Cookie:BIGipServer[poolname]=336268299.20480.0000; expires=Sat, 01-Jan-2000
00:00:00 GMT; path=/
```

Figure 7.3 *Sample cookie template with additional information*

To create your cookie using the sample string above, simply enter the actual pool names and the desired expiration date and time.

merge

```
b [-log] merge [<file_name>]
```

Use the **merge** command to load the BIG-IP configuration from <**file_name**> without resetting the current configuration.

mirror

```
b mirror <mirror_to_if> interfaces add <if_list>
b mirror <mirror_to_if> interfaces delete <if_list>
```

For the BIG-IP Application Switch, you can copy traffic from any port or set of ports to a single, separate port. This is called ***port mirroring***. You should attach a sniffer device to the target port, called the ***mirror-to*** port, for debugging and/or monitoring.

Creating a port mirror

Creating a port mirror consists of specifying a mirror-to port and adding to it one or more ports (that is, a port list) to be mirrored. The **bigpipe** syntax for setting up port mirroring is:

```
b mirror <mirror_to_if> interfaces add <if_list>
```

For example, you could type the following command:

```
b mirror 3.24 interfaces add 3.1 3.3 3.10
```

Deleting interfaces from a port mirror or deleting a port mirror

The **bigpipe** syntax for deleting interfaces from the port mirror is:

```
b mirror <mirror_to_if> interfaces delete <if_list>
```

For example, you could type the following command:

```
b mirror 3.24 interfaces delete 3.10
```

The **bigpipe** syntax for deleting the port mirror is:

```
b mirror <mirror_to_if> delete
```

For example, you could type the following command:

```
b mirror 3.24 delete
```

monitor

```
b monitor <monitor_name> '{ use <monitor_template> [<attr> <attr_value>]... }'
b monitor show [all]
b monitor dump [all]
b monitor <name> show
b monitor <name> delete
b monitor <name> enable | disable
b monitor instance <ip>:<service>  enable | disable
b monitor instance <ip> enable | disable
```

Defines a health monitor. A health monitor is a configuration object that defines how and at what intervals a node is pinged to determine if it is **up** or **down**. Once a monitor is defined, instances of the monitor are created for a node or nodes, one instance per node, using the **bigpipe node** command.

Monitor template attributes

Table 7.3 lists the monitor templates and shows the template-specific attribute sets for each.

Name/Type	Template-Specific Attribute Set
icmp	none
tcp_echo	transparent (optional)
tcp	send "" recv "" transparent (optional) reverse (optional)
http	username "" password "" send "GET /index.html" recv "" get (optional) url (optional) transparent (optional) reverse (optional)
https	username "" password "" send "GET /index.html" recv "" get (optional) url (optional) transparent (optional) reverse (optional)

Table 7.3 *The monitor templates*

Name/Type	Template-Specific Attribute Set
external	run "" args ""
ftp	username "anonymous" password "bigip1@internal" get "/README" url (optional)
nntp	username "" password "" newsgroup "local"
pop3	username "" password ""
smtp	domain "bigip1@internal"
snmp_dca	CPU coefficient "" CPU threshold "" memory coefficient "" memory threshold "" disk coefficient "" disk threshold "" useroid "" useroid coefficient "" useroid threshold ""
snmp_dca_base	useroid "" useroid coefficient "" useroid threshold ""
imap	username "" password "" folder "INBOX" message_num (optional)
radius	username "username" password "password" secret "12345678"
ldap	base "o=Org, c=US" filter "sn=Doe"
sql	username "" password "" database ""
https_443	dest *:443

Table 7.3 *The monitor templates*

Table 7.4 defines the attributes used in the templates.

Attribute	Definition
interval <seconds>	Ping frequency time interval in seconds.
timeout <seconds>	Ping timeout in seconds.
dest <node_addr>	Ping destination node. **<node_address>** Usually ***:*** for simple monitors, ***:*** for all others, causing the monitor instance to ping the address or **address:port** for which it is instantiated. Specifying address and/or port forces the destination to that address/port.
send <string>	Send string for ECV. Default **send** and **recv** values are empty (""), matching any string.
recv <string>	Receive expression for ECV. Default **send** and **recv** values are empty (""), matching any string.
get <string>	For the **http** and **https** monitors **get** replaces the **recv** statement, automatically filling in **"GET".** For the **ftp** monitor **get** can be used to specify a full path to a file. This will automatically fill in **dest**.
url	For the **http**, **https**, and **ftp** monitors, **url** replaces the **recv** statement, supplying a URL and automatically fill in **dest** with the URL address.
reverse	A mode that sets the node **down** if the received content matches the **recv** string.
transparent	A mode that forces pinging through the node to the **dest** address for transparent nodes, such as firewalls.
run <program>	An external user-added EAV program.
args <program_args>	List of command line arguments for external program. **args** are quoted strings set apart by spaces.
username <username>	User name for services with password security. For **ldap** this is a **distinguished** name (an LDAP-format user name).
password <password>	Password for services with password security.
newsgroup <newsgroup>	Newsgroup, for type **nntp** EAV checking only
database <database>	Database name, for type **sql** EAV checking only.
domain <domain_name>	Domain name, for type **smtp** EAV checking only

Table 7.4 *Monitor attributes*

Attribute	Definition
secret	Shared secret for **radius** EAV checking only.
folder	Folder name for **imap** EAV checking only.
message_num	Optional message number for imap EAV checking only
base	Starting place in the LDAP hierarchy from which to begin the query, for **ldap** EAV checking only.
filter	LDAP- format key of what is to be searched for, for **ldap** EAV checking only.

Table 7.4 *Monitor attributes*

-n

```
b -n
```

Use the **-n** option in combination with other commands, such as **bigpipe virtual**, to display services and IP addresses numerically rather than by service name and host name, respectively. For example, type the following command to display services numerically:

```
b -n virtual
```

Figure 7.4 shows an example of output that uses IP address instead of host names.

```
virtual +------> 11.100.1.1          UNIT 1
    |            (cur, max, limit, tot) = (0, 0, 0, 0)
    |            (pckts,bits) in = (0, 0), out = (0, 0)
    +---+--> SERVICE 80                UP
        |        (cur, max, limit, tot) = (0, 0, 0, 0)
        |        (pckts,bits) in = (0, 0), out = (0, 0)
        MEMBER 11.12.1.100:80          UP
                 (cur, max, limit, tot) = (0, 0, 0, 0)
                 (pckts,bits) in = (0, 0), out = (0, 0)
```

Figure 7.4 *The output of* ***bigpipe -n virtual***

nat

```
b nat <orig_addr> to <trans_addr> [unit <unit ID>]
b nat <orig_addr> [...<orig_addr>] delete
b nat [<trans_addr> [...<trans_addr>] ] show | delete
b nat [<orig_addr> [...<orig_addr>] ] show | delete
b nat [<orig_addr>...] stats reset
b nat <orig_addr> vlans <vlan_list> enable | disable
b nat <orig_addr> vlans delete all
b nat <orig_addr> vlans show
b nat <orig_addr> arp [enable | disable | show]
```

Defines an IP address, routable on the external network, that a node can use to initiate connections to hosts on the external network and receive direct connections from clients on the external network. The NAT (Network Address Translation) command defines a mapping between the IP address of a server behind the BIG-IP <**orig_addr**> and an unused routable address on the network in front of the BIG-IP <**trans_addr**>.

node

```
b node <node_ip>[:<service>]... enable | disable
b node <node_ip>[:<service>... show
b node <node_ip>[:<service>]... limit <max_conn>
b node [<node_ip>:<service>]... stats reset
b node <node_ip>[:service] up | down
b node <node_ip>[:<service>] monitor use <monitor_name> [and <monitor_name>]...
b node [<node_ip>[:<service>]] monitor show | delete
b node <node_ip>[<node_ip>]... virtual | actual
```

Displays information about nodes and allows you to set properties for nodes, and node addresses. Nodes may be identified using wildcard notation. Thus ***** represents all nodes on the network, ***.80** represents all port 80 nodes, **11.11.11.1:*** represents all nodes with address **11.11.11.1.**

pool

```
b pool <pool_name> { lb_method <lb_method_specification> <member_definition> }
b pool <pool_name> { lb_method <lb_method_specification> persist_mode
    <persist_mode_specification> <member definition>... }
b pool <pool_name> { lb_method <lb_method_specification> min_active_members <min_value>
    <member definition>... }
b pool <pool_name> { lb_method <lb_method_specification> <member_definition> fallback
    <host> <protocol> <port> <URI path> }
b pool <pool_name> { forward }
b pool <pool_name> add { <member definition>... }
b pool <pool_name> delete { <member definition>... }
b pool <pool_name> modify { [lb_method <lb_method_specification>] [persist_mode
    <persist_mode_specification>] <member definition>... }
b pool <pool_name> { snat disable }
b pool <pool_name> header insert <quoted string>
b pool <pool_name> delete
b pool [<pool_name>] show
b pool <pool_name> lb_method show
b pool <pool_name> persist dump
b pool <pool_name> persist dump mirror
b pool <pool_name> { persist_mode simple | cookie | ssl | sip [sip_timeout <timeout>] |
    sticky | msrdp }
b pool sip dump
b pool <pool_name> sticky clear
b pool <pool_name> stats reset
```

Creates, deletes, modifies, or displays pool definitions. You can use pools to group members together with a common load balancing mode and persistence mode.

Specifying load balancing mode

The load balancing modes are specified as values of the attribute **lb_mode**. The **lb_mode** values are shown in Table 7.5.

Mode Name	lb_mode attribute value
Round Robin	rr or omit **lb_mode** specification
Ratio	ratio
Ratio Member	ratio_member
Fastest	fastest

Table 7.5 *Load balancing modes*

Mode Name	lb_mode attribute value
Fastest Member	fastest_member
Least Connections	least_conn
Least Connections Member	least_conn_member
Observed	observed
Observed Member	observed_member
Predictive	predictive
Predictive Member	predictive_member
Dynamic Ratio	dynamic_ratio

Table 7.5 *Load balancing modes*

For more information about the load balancing modes, refer to *Load balancing method*, on page 4-5.

proxy

```
b proxy <ip>:<service> [unit <id>][{] target <virtual|server>> <ip>:<service>
    [clientss] <enable|disable>
    [[clientssl] key <clientside key file name>]
    [[clientssl] cert <clientside certificate file name>]
    [[clientssl] chain <clientside chain file name>]
    [[clientssl] ca file <clientside CA file name>]
    [[clientssl] ca path <clientside CA path>]
    [[clientssl] client cert ca <clientside client certificate CA file name>]
    [[clientssl] cipher insert [<enable | disable>]

    [[clientssl] client cert insert <([versionnum][serial][sigalg][issuer][validity]
    [subject][subpubkey][x509ext][whole][hash])+|disable>]

    [[clientssl] sessionid insert <([initial][current])+|disable>]
    [[clientssl] ciphers \"quoted string\"]
    [[clientssl] invalid [SSLv2][SSLv3][TLSv1]]
    [[clientssl] client cert <request | require | ignore>]
    [[clientssl] authenticate <once | always>]
    [[clientssl] authetnicate depth <num>]

    [serverssl <enable|disable>]
    [serverssl key <serverside key file name>]
    [serverssl cert <serverside certificate file name>]
    [serverssl chain <serverside chain file name>]
    [serverssl ca file <serverside CA file name>]
    [serverssl ca path <serverside CA path>]
    [serverssl \"quoted string\"]
    [serverssl invalid [SSLv2][SSLv3][TLSv1]]
    [serverssl server cert <require | ignore>]
    [serverssl authenticate depth <num>]
    [akamaize <enable|disable>
    [header insert \"quoted string\"]
    [redirects rewrite <<matching | all> [enable] | [disable>]
    [lasthop pool <none|lasthop pool name>]
    [arp <enable|disable>]
    [vlans <vlan name>[<vlan name>...] disable]
    [}]
b proxy <ip>:<service> unit show
b proxy <ip>:<service> target show
b proxy <ip>:<service> clientssl show
b proxy <ip>:<service> [clientssl] key show
b proxy <ip>:<service> [clientssl] cert show

b proxy <ip>:<service> [clientssl] chain showb proxy <ip>:<service> [clientssl] ca file
show

b proxy <ip>:<service> [clientssl] ca path show
b proxy <ip>:<service> [clientssl] client cert ca show
b proxy <ip>:<service> [clientssl] cipher insert show
b proxy <ip>:<service> [clientssl] sessionid insert show
b proxy <ip>:<service> [clientssl] ciphers show
b proxy <ip>:<service> [clientssl] invalid show
b proxy <ip>:<service> [clientssl] client cert show
b proxy <ip>:<service> [clientssl] authenticate show
b proxy <ip>:<service> [clientssl] authenticate depth show
b proxy <ip>:<service> [clientssl] cache size show
b proxy <ip>:<service> [clientssl] cache timeout show
```

```
b proxy <ip:service> vlans show
b proxy <ip>:<service> serverssl show
b proxy <ip>:<service> serverssl key show
b proxy <ip>:<service> serverssl cert show
b proxy <ip>:<service> serverssl chain show
b proxy <ip>:<service> serverssl ca file show
b proxy <ip>:<service> serverssl ca path show
b proxy <ip>:<service> serverssl ciphers show
b proxy <ip>:<service> serverssl invalid show
b proxy <ip>:<service> serverssl server cert show
b proxy <ip>:<service> serverssl authenticate depth show
b proxy <ip>:<service> akamaize show
b proxy <ip>:<service> header insert show
b proxy <ip>:<service> redirects rewrite show
b proxy <ip>:<service> lasthop pool show
b proxy <ip:service> arp show
b proxy <ip:service> vlans show
b proxy [<ip:service>...] show
```

Creates, deletes, modifies, or displays the SSL or content converter proxy definitions on the BIG-IP. For detailed information about setting up the SSL Accelerator feature, see the ***BIG-IP Solutions Guide,*** *Chapter 9, Configuring an SSL Accelerator*. For detailed information about setting up the content converter feature, see the ***BIG-IP Solutions Guide,*** *Chapter 14, Configuring a Content Converter*.

ratio

```
b ratio [<node_ip>] [node_ip> ...] show
b ratio <node_ip> [<node_ip>...] <weight>
```

For the Ratio load balancing mode, this command sets the weight or proportions for one or more node addresses.

reset

```
b reset
```

Use the following syntax to clear the configuration values and counter values from memory:

```
b reset
```

Use this command with caution. All network traffic stops when you run this command.

Typically, this command is used on a standby BIG-IP prior to loading a new **/config/bigip.conf** file that contains new service enable and timeout values.

For example, you can execute the following commands on a standby BIG-IP:

```
b reset
b load <filename>
```

This sequence of commands ensures that only the values set in the <**filename**> specified are in use.

rule

```
b rule <rule_name> '{ if ( <expression> ) { <if statement> | <use statement> | discard |
<cache statement> | <redirect statement> | <hash statement> <if statement> } [ { else
<statement> } ] [ { else if <statement> } ] }'

b rule <rule_name> '{ discard }'

b rule <rule_name> '{ use <pool_name> }'

b rule <rule_name> '{ cache ( <expression> ) { origin_pool <pool_name> cache_pool <pool_name> [
hot_pool <pool_name> ] [ hot_threshold <hit_rate> ] [ cool_threshold <hit_rate> ] [ hit_period
<seconds> ][ content_hash_size <sets_in_content_hash> ] } }'

b rule <rule_name> '{ redirect <redirect URL> }'

b rule <rule name> '{ hash ( variable ) }'

b rule <rule_name> { if '( <statement> ) { use ( <statement> )' } }

b rule <rule_name> { if '( <statement> )' { use '( <statement> )' } else { '(
    <statement> )' } }

b rule <rule_name> { if '( <statement> )' { use '( <statement> )' } else { '(
    <discard_statement> )' } }

b rule <rule_name> { if '( <statement> )' { use '( <statement> )' } else { '(
    <redirect_statement> )' } }

b rule <rule_name> { if '( <statement> )' { use '( <statement> )' } else { '(
    <cache_statement> )' } }

b rule <rule_name> delete

b rule <rule_name> show
```

Creates, delete, or display the rules on the BIG-IP. Rules allow a virtual server to access any number of pools on the BIG-IP. Based upon a simple or complex expressio,n a pool can be selected through a rule. For more detailed information about using rules, see *Rules*, on page 4-49.

◆ Note

Before you define a rule, you must define the pool or pools that you want the rule to reference.

Creating a rule

Rules are generally added to an existing **bigip.conf** file. Note that the rule body should not be enclosed with single quotes in the **bigip.conf** file. For example:

```
rule cgi_rule {
      if ( http_uri ends_with "cgi" ) { use ( cgi_pool ) }
      else { use ( another_pool ) }
 }
```

Figure 7.5 *A rule typed into the* ***bigip.conf***

In this example, if the **http_uri** string ends with **"cgi"**, then the members of pool **cgi_pool** are used. Otherwise, the members of pool **another_pool** are used.

If the rule is defined on the **bigpipe** command line, you can either surround each pair of parentheses in single quotation marks (’), or place a pair of single quotation marks around the braces. These two methods of defining a rule on the command line are shown as follows:

```
b rule <name> if '{ <if_stmt> | <use_stmt> |  <discard_stmt> |
<redirect_stmt> | <cache> }'
```

Or, you can type the same rule using the following syntax:

```
b rule <name> if { '(<if_stmt>)' | '(<use_stmt>)' |
'(<discard_stmt>)' | '(<redirect>)' | '(<cache>)' }
```

For example:

```
b rule simply_red { if '(client_addr == 10.12.12.10)' { use
'(pool_A80)' } }
```

```
b rule simply_redder '{ if (client_addr == 10.12.12.10) { use
(pool_B80) } }'
```

Associating a rule with virtual server

Associate a rule with a virtual server using the following format:

```
bigpipe virtual 10.20.2.101:http use rule cgi_rule
```

Delete a rule

Delete a rule using the following format:

```
bigpipe rule cgi_rule delete
```

Display rules

Display all rules using the following syntax:

```
bigpipe rule show
```

Or to display a specific rule:

```
bigpipe rule <rule name> show
```

save

```
b save [ <filename> | - ]
b base save [ <filename> | - ]
```

Writes the current BIG-IP configuration settings from memory to the configuration files named **/config/bigip.conf** and **/config/bigip_base.conf**. (**/config/bigip.conf** stores high level configuration settings, such as pools, virtual servers, NATs, SNATs, and proxies. **/config/bigip_base.con**f stores low level configuration settings, like, VLANs, non-floating self IP addresses, and interface settings.)

You can type **b save <filename>**, or a hyphen character (-) in place of a file name, to display the configuration on the standard output device.

```
b [base] save -
```

If you are testing and integrating BIG-IP units into a network, you may want to use multiple test configuration files. Use the following syntax to write the current configuration to a file name that you specify:

```
b [base] save <filename>
```

For example, the following command saves the current configuration from memory to an alternate configuration file named **/config/bigip.conf2**.

```
b save /config/bigip.conf2
```

self

```
b self <addr> vlan <vlan_name | vlangroup_name> [ netmask <ip_mask> ][ broadcast
    <broadcast_addr>] [unit <id>]
b self <addr> floating enable | disable
b self <addr> delete
b self <addr> show
b self show
b self <addr> snat automap enable | disable
```

Defines a self IP address on a BIG-IP or 3-DNS system. A self IP address is an IP address mapping to a VLAN or VLAN group and their associated interfaces on a BIG-IP or 3-DNS system. A one true self IP address is assigned to each interface on the unit as part of first time boot configuration, and also a floating (shared) self IP address for units in a redundant pair. Additional self addresses may be created for health checking, gateway failsafe, routing, or other purposes. These additional self addresses are created using the **self** command.

service

```
b service <service> [<service>...] limit <limit>
b service <service> [<service>...] tcp enable | disable
b service <service> [<service>...] timeout tcp <timeout>
b service <service> [<service>...] udp enable | disable
b service <service> [<service>...] timeout udp <timeout>
b service [<service>... ] show
b service [<service>... ] stats reset
```

Enables and disables network traffic on services, and also sets connection limits and timeouts. You can use port numbers or service names (for example, **www**, **http**, or **80**) for the **<service>** parameter. Note that the settings you define with this command control the service for all virtual servers that use it. By default, all services are disabled.

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

snat

```
b snat map <orig_ip> [...<orig_ip>] to <snat_ip><snat_ip> [unit <unit ID>] [netmask
    <ip>] [arp disable] [vlan <vlan_name_list> disable]
b snat map default to <snat_ip> [unit <unit ID>] [netmask <ip>]
b snat <snat_ip> [...<snat_ip>] delete | show
b snat default delete | show
b snat default dump [verbose]
b snat [<snat_ip> [...<snat_ip>] ] dump [verbose]
b snat globals show
b snat default show
b snat [<snat_ip> [...<snat_ip>] ] show
b snat [<snat_ip> [...<snat_ip>] ] delete
b snat [<snat_ip> [...<snat_ip>] ] arp show
b snat [<orig_ip> [...<orig_ip>] limit <max_conn>
b snat limit <max_conn>
b snat default limit <max conn>
b snat <orig_ip> [...<orig_ip>] mirror enable | disable
b snat default mirror enable | disable
b snat <orig_ip> [...<orig_ip>] timeout tcp | udp <seconds>
b snat default timeout tcp | udp <seconds>
b snat <orig_ip> [...<orig_ip>] stats reset
b snat default stats reset
b snat <orig_ip> [...<orig_ip>]>  disable | enable
b snat <snat_ip> [...<snat_ip>] vlans <vlan_list> disable | enable
b snat <snat_ip> [...<snat_ip>] vlans enable all
b snat <snat_ip> [...<snat_ip>] vlans show
b snat map <vlan_name> to auto
b snat <snat_ip> [...<snat_ip>] arp [enable|disable]
```

Defines one or more addresses that nodes can use as a source IP address when initiating connections to hosts on the external network. Note that clients cannot use SNAT addresses to connect directly to nodes.

stp

```
b stp <stp_name> interfaces add <if_list> | all
b stp <stp_name> hello <interval>
b stp <stp_name> max_age <interval>
b stp <stp_name> forward_delay <interval>
b stp <stp_name> interfaces delete <if _list>
b stp <stp_name> enable|disable
```

The BIG-IP IP Application Switch provides Spanning Tree Protocol (STP) implementation for loop resolution in configurations where one or more external switches is connected in parallel with the BIG-IP. This feature allows you to configure two or more interfaces on the platform as an STP domain. For interfaces in the STP domain, the spanning tree algorithm identifies the most efficient path between the network segments, and establishes the switch associated with that path as the **root**. Links forming redundant paths are shut down, to be re-activated only if the root fails.

The STP domain should contain all ports that are connected in parallel to an external switch where there are nodes on the link capable of generating or receiving traffic. You will want a second domain if there is an additional switch or switches connected in parallel with additional BIG-IP interfaces.

summary

```
b summary
```

Displays a summary of current usage statistics. The output display format for the **summary** command is shown in Figure 7.6. You can find detailed descriptions of each of statistic displayed by the **summary** command in *Monitoring the BIG-IP*, on page 11-2.

```
BIG-IP total uptime          = 1 (day) 4 (hr) 40 (min) 8 (sec)
BIG-IP total uptime (secs)   = 103208
BIG-IP total # connections   = 0
BIG-IP total # pkts          = 0
BIG-IP total # bits          = 0
BIG-IP total # pkts(inbound)  = 0
BIG-IP total # bits(inbound)  = 0
BIG-IP total # pkts(outbound) = 0
BIG-IP total # bits(outbound) = 0
BIG-IP error no nodes available         = 0
BIG-IP tcp port deny                    = 0
BIG-IP udp port deny                    = 0
BIG-IP virtual tcp port deny               = 0
BIG-IP virtual udp port deny                = 0
BIG-IP max connections deny             = 0
BIG-IP virtual duplicate syn ssl            = 0
BIG-IP virtual duplicate syn wrong dest     = 0
BIG-IP virtual duplicate syn node down      = 0
BIG-IP virtual maint mode deny              = 0
BIG-IP virtual addr max connections deny = 0
BIG-IP virtual path max connections deny = 0
BIG-IP virtual non syn                      = 0
BIG-IP error not in out table           = 0
BIG-IP error not in in table            = 0
BIG-IP error virtual fragment no port       = 0
BIG-IP error virtual fragment no conn       = 0
BIG-IP error standby shared drop        = 0
BIG-IP dropped inbound                  = 0
BIG-IP dropped outbound                 = 0
BIG-IP reaped                           = 0
BIG-IP ssl reaped                       = 0
BIG-IP persist reaped                   = 0
BIG-IP udp reaped                       = 0
BIG-IP malloc errors                    = 0
BIG-IP bad type                         = 0
BIG-IP mem pool total 96636758 mem pool used 95552 mem percent used   0.10
```

Figure 7.6 *The **summary** output display*

trunk

```
b trunk <controlling_if> define <if_list>
b trunk [<controlling_if>] show [verbose]
b trunk [<controlling_if>] stats reset
```

The **trunk** command aggregates links (individual physical interfaces) to form a trunk. This link aggregation increases the bandwidth of the individual NICs in an additive manner. Thus, four fast Ethernet links, if aggregated, create a single 400 Mb/s link. The other advantage of link aggregation is link failover. If one link in a trunk goes down, traffic is simply redistributed over the remaining links.

A trunk must have a controlling link and acquires all the attributes of that controlling link from Layer 2 and above. Thus, the trunk automatically acquires the VLAN membership of the controlling link but does not acquire its media type and speed. Outbound packets to the controlling link are load balanced across all of the known-good links in the trunk. Inbound packets from any link in the trunk are treated as if they came from the controlling link.

A maximum of eight links may be aggregated. For optimal performance, links should be aggregated in powers of two. Thus ideally you will aggregate two, four, or eight links. Gigabit and fast ethernet links cannot be placed in the same trunk.

For more information on interface naming, refer to *Interface naming conventions*, on page 3-2.

unit

```
b unit [show]
b unit peer [show]
```

The unit number on a system designates which virtual servers use a particular unit in an active-active redundant configuration. You can use the **bigpipe unit** command to display the unit number assigned to a particular BIG-IP. For example, to display the unit number of the unit you are on, type the following command:

```
b unit show
```

To display the unit number of the other unit in a redundant system, type in the following command:

```
b unit peer show
```

Note

If you use this command on a redundant system in active/standby mode, the active unit shows as unit 1 and 2, and the standby unit has no unit numbers.

Tip

*The **bigpipe unit peer show** command is the best way to determine whether the respective state mirroring mechanisms are connected.*

verbose

```
b verbose virtual_server_udp_port_denial
b verbose virtual_server_tcp_port_denial
b verbose bigip_udp_ort_denial
b verbose bigip_tcp_port_denial
```

Used to modify the verbose log level. This command is an alternative to using the **bigpipe global verbose** command.

Table 7.6 defines the command and shows the equivalencies.

b verbose command	b global verbose command
b verbose bigip_udp_port_denial Turns UDP port denial logging **on**. This logs UDP port denials to the BIG-IP address.	**b global verbose_log_level=1**
b verbose bigip_tcp_port_denial Turns TCP port denial logging **on**. This logs TCP port denials to the BIG-IP address.	**b global verbose_log_level=2**
b verbose virtual_server_udp_port_denial Turns virtual UDP port denial logging **on**. This logs UDP port denials to the virtual server address.	**b global verbose_log_level=4**
b verbose virtual_server_tcp_port_denial Turns virtual TCP port denial logging **on**. This logs TCP port denials to the virtual server address.	**b global verbose_log_level=8**
b verbose bigip_udp_port_denial **b verbose bigip_tcp_port_denial** **b verbose bigip_udp_ort_denial** **b verbose bigip_tcp_port_denial** Turns UDP and TCP port denial on for both virtual server and BIG-IP addresses.	**b global verbose_log_level=15**

***Table 7.6 bigpipe verbose** and **global verbose** command equivalencies*

verify

```
b [log] verify <command...]
verify load [<filename> | -]
```

Parses the command line and checks syntax without executing the specified command. This distinguishes between valid and invalid commands

Use the **verify** command followed by a command that you want to validate:

```
b verify virtual 10.10.10.100:80 use pool my_pool
```

The command checks the syntax and logic, reporting any errors that would be encountered if the command executed.

Use the **verify** command together with the **load <filename>** command to validate the specified configuration file. For example, to check the syntax of the configuration file **/config/altbigpipe.conf**, use the following command:

```
b verify load /config/altbigip.conf
```

version

```
b version
```

Displays the version of the BIG-IP operating system and the features enabled.

For example, for a BIG-IP HA, the **bigpipe version** command displays the output shown in Figure 7.7.

```
Product Code:
BIG-IP HA

Enabled Features:
SSL Gateway                             Gateway Failsafe
Static Load Balancing                   Snat
Nat                                     Pools
Akamaizer                               Full Proxy
Late Binding                            HTTP Rules
Mirroring                               Failover
Node HA                                 Dynamic Load Balancing
Destination Address Affinity            Cookie Persistence
SSL Persistence                         Simple Persistence
EAV                                     ECV SSL
ECV                                     ECV Transparent
Health Check                            Filter
```

Figure 7.7 *The **version** output display*

virtual

```
b virtual <virt_ip>[:<service>] [unit <ID>] [netmask <ip>] [broadcast <ip>] use pool
    <pool_name>
b virtual <virt_ip>:<service> [/<bitmask>][unit <ID>] use pool <pool_name>
b virtual <virt_ip>[:<service>] [unit <ID>] [netmask <ip>] use rule <rule_name>
b virtual <virt_ip>[:<service>] [unit <ID>] [netmask <ip>] forward
b virtual <virt_ip>:<service> translate port enable | disable | show
b virtual <virt_ip>:<service> svc_down_reset enable | disable | show
b virtual <virt_ip>:<service> translate addr enable | disable | show
b virtual <virt_ip>:<service> lasthop pool <pool_name> | none | show
b virtual <virt_ip>:<service> mirror conn enable | disable | show
b virtual <virt_ip>:<service> conn rebind enable | disable | show
b virtual [<virt_ip:service>] stats reset
b virtual <virt_ip>:<service> accelerate enable | disable | show
b virtual <virt_ip>:<service> use pool <pool_name> accelerate disable
b virtual <virt_ip>:<service> vlans <vlan_list> disable | enable
b virtual <virt_ip>:<service> vlans show
b virtual <virt_ip> arp enable|disable|show
b virtual <virt_ip> any_ip enable | disable
b virtual <virt_ip> any_ip timeout <seconds>
b virtual <virt_ip> [:<service>] [...<virt_ip>[:<service>]] show
b virtual <virt_ip> [:<service>] [...<virt_ip>[:<service>]] enable|disable
b virtual <virt_ip>[:<service>] [ ... <virt_ip>[:<service>]] delete
b virtual <virt_ip>[:<service>] [... <virt_ip>[:<service>]] limit <max_conn>
b virtual <vlan_name>[:service>]
b virtual <vlan_name> use pool <pool_name>
```

Creates, deletes, and displays information about virtual servers. This command also sets connection mirroring, connection limits, and timeouts on a virtual server.

vlan

```
b vlan <name> rename <new_name>
b vlan <vlan_name> delete
b vlan <vlan_name> tag <tag_number>
b vlan <vlan_name> interfaces add [tagged] <if_list>
b vlan <vlan_name> interfaces delete <if_list>
b vlan <vlan_name> interfaces delete all
b vlan <vlan_name> interfaces show
b vlan <vlan_name> port_lockdown  enable | disable
b vlan <vlan_name> bridging enable | disable
b vlan <vlangroup_name> proxy_forward enable | disable
b vlan <vlan_name> failsafe arm | disarm | show
b vlan <vlan_name> timeout <seconds> | show
b vlan <vlan_name> snat automap
b vlan show
b vlan <vlan_name> show
b vlan <vlan_name> interfaces show
b vlan <vlan_name> rename <new_vlan_name>
b vlan <if_name> mac_masq <mac_addr> | show
b vlan <if_name> mac_masq 0:0:0:0:0
vlan <vlan name> l2_agingtime <seconds>
vlan <vlan name> fdb add <MAC address> interface <ifname>
vlan <vlan name> fdb delete <MAC address> interface <ifname>
vlan <vlan name> fdb static show
vlan <vlan name> fdb dynamic show
vlan <vlan name> fdb show
```

The **vlan** command defines VLANs, VLAN mappings, and VLAN properties. By default, each interface on a BIG-IP or 3-DNS system is an untagged member of a VLAN. The lowest-numbered interface is assigned to the **external** VLAN, the interface on the main board is assigned to the **admin** VLAN, and all other interfaces are assigned to the **internal** VLAN.

If the tag number specified when defining a VLAN is 0, the **vlan** command creates an empty VLAN.

vlangroup

```
vlangroup [<vlan name list>] [show]
vlangroup [<vlan name list>] list
vlangroup <vlan name list> delete
vlangroup <vlan name> tag <number>
vlangroup [<vlan name list>] tag [show]
vlangroup [<vlan name list>] interfaces [show]
vlangroup <vlan name> vlans add <vlan if name list>
vlangroup <vlan name list> vlans delete <vlan if name list>
vlangroup <vlan name list> vlans delete all
vlangroup [<vlan name list>] vlans [show]
vlangroup <vlan name list> port_lockdown enable | disable
vlangroup [<vlan name list>] port_lockdown [show]
vlangroup <vlan name list> proxy_forward enable | disable
vlangroup [<vlan name list>] proxy_forward [show] vlangroup <vlan name list> failsafe
    arm
vlangroup <vlan name list> failsafe disarm
vlangroup [<vlan name list>] failsafe [show]
vlangroup <vlan name list> timeout <number>
vlangroup [<vlan name list>] timeout [show] vlangroup <vlan name list> snat automap
    enable (deprecated)
vlangroup <vlan name list> snat automap disable (deprecated)
vlangroup <vlan name list> mac_masq <MAC addr>
vlangroup [<vlan name list>] mac_masq [show]
vlangroup <vlan name list> fdb add <MAC addr> interface <if name>
 vlangroup <vlan name list> fdb delete <MAC addr> interface <if name>
vlangroup [<vlan name list>] fdb [show]
vlangroup [<vlan name list>] fdb show static
vlangroup [<vlan name list>] fdb show dynamic
vlangroup <vlan name> rename <vlan name>
```

The **vlangroup** command defines a VLAN group, which is a grouping of two or more VLANs belonging to the same IP network for the purpose of allowing L2 packet forwarding between those VLANs.

The VLANs between which the packets are to be passed must be on the same IP network, and they must be grouped using the **vlangroup** command. For example:

```
b vlangroup network11 { vlans add internal external }
```

A self IP address must be assigned to the VLAN group using the following command:

```
b self <ip_addr> vlan network11
```

L2 forwarding must be enabled for the VLAN group using the **VLAN proxy_forward** attribute. This attribute is enabled by default when the VLAN group is enabled.

8

Configuring SNMP

- Introduction
- Downloading the MIBs
- Configuring SNMP using the Configuration utility
- SNMP configuration files
- Configuring snmpd to send responses out of different ports or addresses

Introduction

This chapter covers the management and configuration tasks for the simple network management protocol (SNMP) agent and management information bases (MIBs) available with the BIG-IP.

◆ Note

On a BIG-IP with a 3-DNS module installed, you must configure the SNMP agent in order to use the SEE-IT Network Manager.

The BIG-IP SNMP agent and MIBs allow you to manage the BIG-IP by configuring traps for the SNMP agent or polling the BIG-IP with your standard network management station (NMS).

You can use the Configuration utility to configure the BIG-IP SNMP agent to send traps to your management system. You can also set up custom traps by editing several configuration files.

You can use SNMP security options to securely manage access to information collected by the BIG-IP SNMP agent, including Community names, TCP wrappers, and View Access Control Mechanism (VACM).

This chapter is divided into three parts:

- **Downloading the MIBS**
  This section shows how to download the SNMP MIBs.
- **Configuring SNMP using the Configuration utility**
  This section shows how to set up SNMP for a remote administrative host.
- **SNMP configuration files**
  This section describes the SNMP configuration files and their syntax.
- **Configuring snmpd to respond out of different ports and addresses**
  This section describes how to configure **snmpd** to respond out of different ports and addresses

Downloading the MIBs

To set up SNMP for a remote network management station, you must download and install the product-specific MIB files. For all BIG-IP units there are the following product-specific MIB files:

- **LOAD-BAL-SYSTEM-MIB.txt**.
  This is an enterprise MIB that contains specific information for properties associated with specific BIG-IP functionality (load balancing, NATs, and SNATs).
- **UCD-SNMP-MIB.txt**.
  This is an enterprise MIB that contains information and metrics about memory, disk utilization and other information regarding the BIG-IP operating system. It is fully documented in RFC 1213.

- **Etherlike-MIB.txt**
  This is a standard MIB which describes statistics for the collection of ethernet interfaces attached to the system. It is fully documented in RFC-2665.
- **If-MIB.txt**
  This MIB supports an extended version of the **ifTable** including 64-bit counters.
- **RMON-MIB.txt**
  This is a standard MIB that describes real-time and historical statistics for the ethernet systems in the interface. This MIB also allows the setting of alerts and traps based on user defined thresholds of available metrics in the system. It is fully documented in RFC 2819s.
- **rfc1525.mib**
  This is a standard MIB which describes objects for managing MAC bridges based on the IEEE 802.1D-1990 standard between Local Area Network (LAN) segments. It is fully documented in RFCs 1463 and 1525.

For a BIG-IP with the 3-DNS module there are two additional product-specific MIB files:

- **RFC1611.my**
  This is the DNS MIB (for the 3-DNS module only).
- **3dns.my**
  This is an enterprise MIB which describes information and properties of objects associated with the functioning of 3-DNS (for the 3-DNS module only).

You can download these files from the **Additional Software Downloads** section of the Configuration utility home page, where they appear as the following hypertext entries:

- **BIG-IPMIB** (LOAD-BAL-SYSTEM-MIB.txt and UCD-SNMP-MIB.txt)
- **Interface MIB** (If-MIB.txt)
- **RMON MIB** (RMON-MIB.tx)
- **BRIDGE** (rfc1525.mib)

You can also download these files directly from **/usr/local/share/snmp/mibs** on the BIG-IP to your remote host using **ssh** and **scp** (crypto version) **telnet** and **ftp** (non-crypto version).

Configuring SNMP using the Configuration utility

To configure SNMP for a remote network management station, you must perform the following tasks:

- **Set up client access**
  Configure the BIG-IP to allow administrative access to the SNMP agent.
- **Configure system information**
  Set the system information variables.
- **Configure Traps**
  Enable traps and specify by community, port, and sink.

All three tasks are performed using the SNMP Administration screen, shown in Figure 8.1. To access this screen, simply click **System Admin** in the navigation pane, then click the SNMP Administration tab.

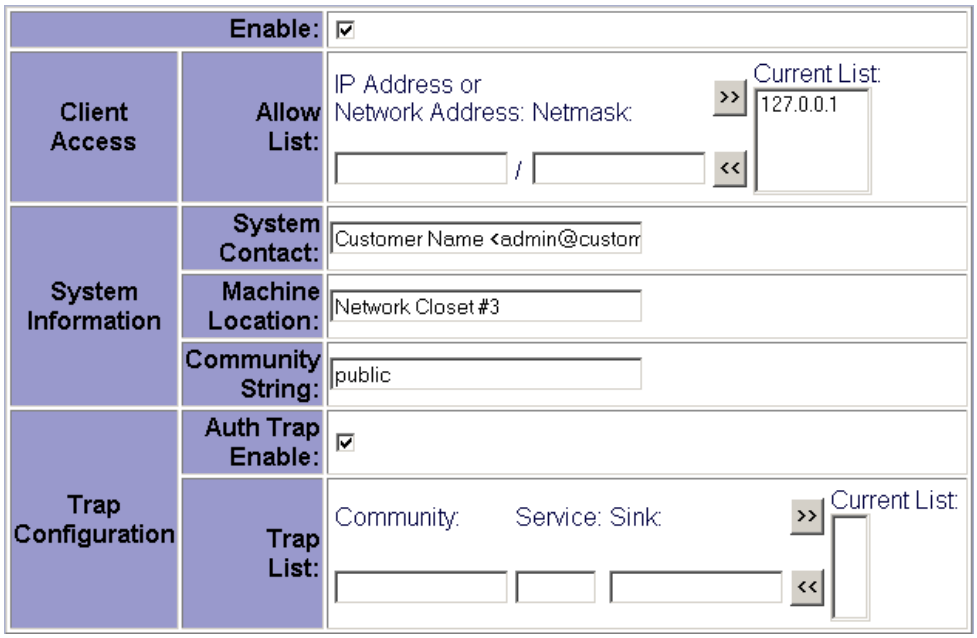

Figure 8.1 *SNMP Administration screen*

Setting up client access

To set up client access, you enable access and specify the IP or network addresses (with netmasks as required) from which the SNMP agent can accept requests. (By default, SNMP is enabled only for the BIG-IP loopback interface **127.0.0.1**.)

To allow access to the SNMP agent using the Configuration utility

1. In the top of the SNMP Administration screen, check the **Enable** box to allow access to the BIG-IP SNMP agent.
2. In the Client Access Allow List section, type the following information:
   - **IP Address or Network Address**
     Type in an IP address or network address from which the SNMP agent can accept requests. Click the Add (>>) button to add the address to the Current List. For a network address, type in a netmask.
   - **Netmask**
     If you type a network address in the **IP Address or Network Address** box, type the netmask for the network address in this box.
3. Click the Add (>>) button to add the network address to the **Current List**.

Configuring system information

System information includes certain traps, passwords, and general SNMP variable names. There are three main variables:

- **System Contact name**
  The System Contact is a MIB-II simple string variable defined by almost all SNMP boxes. It usually contains a user name, as well as an email address.
- **Machine Location (string)**
  The Machine Location is a MIB-II variable that almost all boxes support. It is a simple string that defines the location of the box.
- **Community String**
  The community string clear text password is used for basic SNMP security. This also maps to VACM groups, but for initial read/only access, it is limited to just one group.

To set system information properties using the Configuration utility

You use the System Information section of the SNMP Administration screen to set the system information properties.

1. In the **System Contact** box, type the contact name and email address for the person to contact regarding issues with this BIG-IP.
2. In the **Machine Location** box, type a machine location, such as First Floor, or Building 1, that describes the physical location of the BIG-IP.

3. In the **Community String** box, type a community name. The community name is a clear text password used for basic SNMP security and for grouping machines that you manage.

Configuring traps

To configure traps, you provide three pieces of information:

- **trapcommunity <community string>**
  This sets the community string (password) to use for sending traps. If set, it also sends a trap upon startup: **coldStart(0)**.
- **trapport <port>**
  This sets the port on which traps are sent. There must be one **trapport** line for each **trapsink** host.
- **authtrapenable <integer>**
  Setting this variable to **1** enables traps to be sent for authentication warnings. Setting it to **2** disables it.

To set trap configuration properties using the Configuration utility

You use the Trap Configuration section of the SNMP Administration screen to set trap properties.

1. Check the **Auth Trap Enabled** box to allow traps to be sent for authentication warnings.
2. In the **Community** box, type the community name to which this BIG-IP belongs. Traps sent from this box are sent to the management system managing this community.
3. In the **Service** box, type the service name on which the BIG-IP sends traps. Traps sent from the BIG-IP are sent to the management system on through this port.
4. In the **Sink** box, type the host that should be notified when a trap is sent by the BIG-IP SNMP agent.
5. Click the Add (>>) button to add it to the **Current List**. To remove a trap sink from the **Current List**, click the trap sink you want to remove, and click the Remove (<<) button.
6. Click the **Apply** button.

SNMP configuration files

The SNMP options that you specify in the SNMP Administration screen are written to one or more of the following configuration file or files. If you prefer, you can configure SNMP by directly editing the appropriate files with a text editor rather than using the Configuration utility.

- **hosts.deny**
  This file denies all UDP connections to the SNMP agent.
- **hosts.allow**
  This file specifies which hosts are allowed to access the SNMP agent.
- **snmpd.conf**
  This file configures the SNMP agent.
- **snmptrap.conf**
  For the BIG-IP, the configuration in **/etc/snmptrap.conf** determines which messages generate traps, and what those traps are. Edit this file only if you want to add traps.
- **3dns_snmptrap.conf**
  For the 3-DNS Controller, the configuration in **/etc/3dns_snmptrap.conf** determines which messages generate traps and what those traps are. Edit this file only if you want to add traps.
- **syslog.conf**
  Configure **/etc/syslog.conf** to pipe specified message types through **checktrap.pl**.

/etc/hosts.deny

This file must be present to deny by default all UDP connections to the SNMP agent. The contents of this file are as follows:

```
ALL : ALL
```

/etc/hosts.allow

The **/etc/hosts.allow** file is used to specify which hosts are allowed to access the SNMP agent. There are two ways to configure access to the SNMP agent with the **/etc/host.allow** file. You can type in an IP address, or list of IP addresses, that are allowed to access the SNMP agent, or you can type in a network address and mask to allow a range of addresses in a subnetwork to access the SNMP agent.

For a specific list of addresses, type in the list of addresses you want to allow to access the SNMP agent. Addresses in the list must be separated by blank space or by commas. The basic syntax is as follows:

```
daemon: <IP address> <IP address> <IP address>
```

For example, you can type the following line which sets the SNMP agent to accept connections from the IP addresses specified:

```
bigsnmpd: 128.95.46.5 128.95.46.6 128.95.46.7
```

For a range of addresses, the basic syntax is as follows, where **daemon** is the name of the daemon, and **IP/MASK** specifies the network that is allowed access. The **IP** must be a network address:

```
daemon: IP/MASK
```

For example, you might use the following line which sets the **bigsnmpd** daemon to allow connections from the **128.95.46.0/255.255.255.0** address:

```
bigsnmpd: 128.95.46.0/255.255.255.0
```

The example above allows the 254 possible hosts from the network address **128.95.46.0** to access the SNMP daemon. Additionally, you may use the keyword **ALL** to allow access for all hosts or all daemons.

◆ Note

192.168.1/24 CIDR syntax is not allowed.

/etc/snmpd.conf

The **/etc/snmpd.conf** file controls most of the SNMP agent. This file is used to set up and configure certain traps, passwords, and general SNMP variable names. A few of the necessary variables are listed below:

- **System Contact Name**
  The System Contact is a MIB-II simple string variable defined by almost all SNMP boxes. It usually contains a user name, as well as an email address. This is set by the **syscontact** key.
- **Machine Location (string)**
  The Machine Location is a MIB-II variable that almost all boxes support. It is a simple string that defines the location of the box. This is set by the **syslocation** key.
- **Community String**
  The community string clear text password is used for basic SNMP security. This also maps to VACM groups, but for initial read/only access it is limited to only one group.
- **Trap Configuration**
  Trap configuration is controlled by these entries in the **/etc/snmpd.conf** file:
  - **trapsink <host>**
    This sets the host to receive trap information. The **<host>** is an IP address.
  - **trapport <port>**
    This sets the port on which traps are sent. There must be one **trapport** line for each **trapsink** host.

- **trapcommunity <community string>**
  This sets the community string (password) to use for sending traps. If set, it also sends a trap upon startup: **coldStart(0)**.
- **authtrapenable <integer>**
  Setting this variable to **1** enables traps to be sent for authentication warnings. Setting it to **2** disables it.
- **data_cache_duration <seconds>**
  This is the time in seconds during which data is cached. The default value for this setting is one second.

Note

*A **trapport** line controls all **trapsink** lines that follow it until another **trapport** line appears. Therefore, to change the trap port for a trap sink, the new **trapport** line must be inserted before the trap sink's **trapsink** line, with no other **trapport** lines in between. The same logic follows for **trapcommunity** lines.*

/etc/snmptrap.conf

This configuration file includes OID, trap, and regular expression mappings. The configuration file specifies whether to send a specific trap based on a regular expression. An excerpt of the configuration file is shown in Figure 8.2.

```
# Default traps.
.1.3.6.1.4.1.3375.1.1.110.2.6 (ROOT LOGIN) ROOT LOGIN
.1.3.6.1.4.1.3375.1.1.110.2.5 (denial) REQUEST DENIAL
.1.3.6.1.4.1.3375.1.1.110.2.4 (BIG-IP Loading) SYSTEM RESET
.1.3.6.1.4.1.3375.1.1.110.2.3 (Service detected UP) SERVICE UP
.1.3.6.1.4.1.3375.1.1.110.2.2 (Service detected DOWN) SERVICE DOWN
#.1.3.6.1.4.1.3375.1.1.110.2.1 (error) Unknown Error
#.1.3.6.1.4.1.3375.1.1.110.2.1 (failure) Unknown Failure
```

Figure 8.2 *Excerpt from the **/etc/snmptrap.conf** file*

Some of the OIDs have been permanently mapped to BIG-IP specific events. The OIDs that are permanently mapped for the BIG-IP include:

- Root login
- Request denial
- System reset
- Service up
- Service down

You may, however, insert your own regular expressions and map them to the **110.1 OID**. The **/etc/snmptrap.conf** file contains two examples for mapping your own OIDs:

- Unknown error
- Unknown failure

By default, the lines for these files are commented out. Use these OIDs for miscellaneous events. When lines match your expression, they are sent to your management software with the 110.2.1 OID.

If you change this file, restart the SNMP agent **bigsnmpd** as follows:

```
bigstart restart bigsnmpd
```

For the 3-DNS Controller, the configuration in **/etc/3dns_snmptrap.conf** determines which messages generate traps and what those traps are. Edit this file only if you want to add traps.

Syslog

In order to generate traps, you must configure **syslog** to send syslog lines to **checktrap.pl**. If the syslog lines make a match to the specified configuration in the **snmptrap.conf** file, a valid SNMP trap is generated. The following lines in the **/etc/syslog.conf** file require that the **syslog** examine information logged, scan the **snmptrap.conf** file, and determine if a trap should be generated:

```
local0.* | exec /sbin/checktrap.pl.
local1.* | exec /sbin/checktrap.pl.
auth.* | exec /sbin/checktrap.pl.
local2.* | exec /sbin/checktrap.pl.
```
(for 3-DNS only)

◆ Note

*If you uncomment these lines, make sure you restart **syslogd**.*

If you change this file, restart the SNMP agent **bigsnmpd** with the following command:

```
bigstart restart bigsnmpd
```

Configuring snmpd to send responses out of different ports or addresses

You can configure the **snmpd** to respond on different ports or bind the daemon to a specific interface. Use the following syntax to configure **snmpd**:

```
snmpd -p [(udp|tcp):]port[@address][,...]
```

Use this command to make the agent list on the specified list of sockets instead of the default port, which is port 161. Separate multiple ports by commas. You can specify transports by prepending the port number with the transport name (**udp** or **tcp**) followed by a colon.

To bind to a particular interface, you can specify the address you want it to bind with. For example, you can specify the following command to make the agent listen on UDP port 161 for any address, TCP port 161 for any address, and UDP port 9161 on only the interface associated with the localhost address.

```
snmpd -p 161,tcp:161,9161@localhost
```

◆ Note

*The **-T** flag changes the default transport mapping to use (in the previous example, the default transport mapping is UDP).*

9

BIG/db Configuration Keys

- Supported BIG/db configuration keys

Supported BIG/db configuration keys

The BIG/db is a database that contains configuration elements for the BIG-IP. Configuration options that BIG/db supports include:

- Fail-over
- State mirroring
- Gateway failsafe pingers
- Configuration synchronization
- Interface related settings
- Health monitor settings

The BIG/db keys for each of these features are described in the following series of tables. The keys are viewed and set using the **bigpipe db** command.

```
b db get <key>
b db get <reg_exp>
b db set <key>
b db set <key> = <value>
b db unset <key>
b db unset <reg_exp>
b db dump [filename]
```

To display current setting of a BIG/db configuration key

To display the value of a BIG/db configuration key, use the following syntax:

```
b db get <key>
b db get <regular_exp>
```

For example, the following command displays the value of **Local.Bigip.FTB.HostNumber**:

```
b db get Local.Bigip.FTB.HostNumber
```

The following command displays the value of all local keys:

```
b db get Local.*
```

To set a BIG/db configuration key

To create (set) a BIG/db configuration key, use the following syntax:

```
b db set <key>
```

To set a BIG/db configuration key and assign a value to it, use the following syntax:

```
b db set <key> = <value>
```

For example, the following command sets **Local.Bigip.FTB.HostNumber** mode to **on**:

```
b db set Local.Bigip.FTB.HostNumber = 1
```

To unset a BIG/db configuration key

To unset a BIG/db configuration key, use the following syntax:

```
b db unset <key>
b db unset <regular_exp>
```

For example, the following command unsets **Local.Bigip.FTB.HostNumber**:

```
b db unset Local.Bigip.FTB.HostNumber
```

The following command unsets all local keys:

```
b db unset set Local.*
```

Failover and cluster keys

The failover and cluster keys (Table 9.1) control failover from the active to the standby unit in a BIG-IP redundant system. If you change one of these values, you must update the BIG-IP configuration with the **bigpipe failover init** command (which is the same as **bigstart reinit sod)**. This command forces the system to reread the BIG/db database and use the new values. To run this command, type the following:

```
b failover init
```

Fail-Over Key Name	Description
Common.Bigip.Failover.AwaitPeerAliveDelay = 2	Delay in seconds before testing whether peer is active. The default value is **2**.
Common.Bigip.Failover.AwaitPeerDeadDelay = 1	Delay in seconds before testing whether the peer has failed. The default value is **1**.
Common.Bigip.Failover.FailbackDelay= 60	Use active-active mode if set to one. By default this is off and active-standby mode is used.
Local.Bigip.Failover.UnitId	This key is required. Each BIG-IP must have a unique unit ID of **1** or **2** in the event that network communication is not possible with its peer.
Common.Bigip.Failover.ActiveMode = 0	Use active-active mode if set to **1**. By default, this is off and active/standby mode is used.
Common.Bigip.Failover.ManFailBack = 0	If using active-active mode, the fail-over mechanism waits until receiving a command before surrendering resources to a rebooted machine.
Common.Bigip.Failover.NoSyncTime	By default, one BIG-IP synchronizes its time with the other. Set this key to **1** to turn off the time synchronization feature.
Common.Bigip.Failover.DbgFile	File into which **sod** logs the fail-over debug information.

Table 9.1 *The BIG/db keys that store fail-over information*

Fail-Over Key Name	Description
Common.Bigip.Failover.PrintPeerState = 0	The default value for this key is **0**. Fail-over daemon (**/sbin/sod**) writes the state of its connection to its peer, hardwire and/or network. This information is written to the fail-over daemon's debug log file.
Common.Bigip.Failover.UseTty00 = 0	Failover daemon uses **/dev/tty00** for hardwired failover.
Common.Bigip.Failover.UseTty01 = 1	Failover daemon uses **/dev/tty01** for hardwired failover.
Local.Bigip.Failover.ForceActive = 0	Failover daemon always attempts to become the active unit.
Local.Bigip.Failover.ForceStandby = 0	Failover daemon goes to standby whenever it senses that its peer is alive.
Common.Sys.Failover.Network = 0	Use the network by way of the **sfd** as a backup to, or instead of, the serial line for fail-over if this value is **1**. By default, this feature is off.
Common.Bigip.Cluster.ActiveKeepAliveSec = 1	The default value for this key is **0**. An active unit sends a heartbeat message to its peer with this frequency. Default is **1** second.
Common.Bigip.Cluster.StandbyTimeoutSec = 3	Consider the peer BIG-IP failed if no message is received within the timeout period. Used by network fail-over. Default is **3** seconds.

Table 9.1 *The BIG/db keys that store fail-over information*

StateMirror keys

StateMirror keys (Table 9.2) control state mirroring. If you change one of these values, you must perform the following reinitialization:

```
bigstart reinit sfd
```

State Mirroring Key Name	Description
Common.Bigip.StateMirror.DbgFile	File into which debug information is written, required if debug level is specified (below).
Common.Bigip.StateMirror.DbgLevel = 0	The debug level is composed of the following options: 1 - log reads and writes 2 - log connection attempts and results 4 - log state changes 8 - open log files in append mode, the default is to truncate the files when opening. The debug level is **0** by default.

Table 9.2 *The BIG/db keys that store state mirroring information*

State Mirroring Key Name	Description
Common.Bigip.StateMirror.NoGC = 0	By default, state mirroring causes mirrored data structures to be deleted when it receives a new connection. This key is brought up to date by the unit's peer. This can cause a delay if the system is absolutely loaded. Turning off the GC is provided as an option.
Common.Bigip.StateMirror.ActiveFile	Enables writing of data from the active unit's kernel into the ActiveFile file. This data file can be read with the **sfread** program.
Common.Bigip.StateMirror.StandbyFile	Enables writing of standby data into the StandBy file. This data file can be read with the **sfread** program.
Common.Bigip.StateMirror.Username	IP address of this BIG-IP. By default **sfd** uses the first internal interface in the list returned by the **bigapi_quer_children BIGapi** command. Set this when using multiple NICs.
Common.Bigip.StateMirror.PeerListenPort = 1028	Port on which the BIG-IP listens for connections from the active unit. The default is **1028**.
Local.Bigip.StateMirror.Ipaddr	IP address of this BIG-IP.
Local.Bigip.StateMirror.PeerIpaddr	IP address of the BIG-IP peer unit. A value is required.

Table 9.2 *The BIG/db keys that store state mirroring information*

Using Gateway Pinger keys

The **GatewayPinger** keys (Table 9.3) control the gateway failsafe pinger. If you change one of these values, you must reinitialize the system as follows:

```
bigstart reinit bigd
```

The following table lists the Gateway Pinger keys.

Gateway Pinger Key Name	Description
Local.Bigip.GatewayPinger.Ipaddr = 0.0.0.0 Local.Pinger.alias=Local.Bigip.GatewayPinger.Ipaddr	IP address or host name of the gateway router for the BIG-IP
Local.Bigip.GatewayPinger.Pinginterval = 0	Ping interval of this BIG-IP gateway pinger
Local.Bigip.GatewayPinger.Timeout = 0	Timeout of the BIG-IP gateway pinger

Table 9.3 *The BIG/db keys that store gateway pinger information*

Bigd keys

The **Bigd** keys (Table 9.4) control the health monitors. If you change one of these values, you must re-initialize the system as follows:

```
bigstart reinit bigd
```

Bigd Key Name	Description
Common.Bigip.Bigd.Verbose = 0	Set to non-zero to cause bigd to generate output to debug file.
Common.Bigip.Bigd.SimulatePings = 0	Set to non-zero to cause **bigd** to generate pings but not report results to the kernel.
Common.Bigip.Bigd.RecvMatchAll = 0	Set to non-zero to cause **bigd** to allow any response from the node as a receive match.
Common.Bigip.Bigd.NodePingOff = 0	Set to non-zero to turn off (noisy) **bigd** node pings. Service pings are still enabled.
Common.Bigip.Bigd.NodePingTcp = 0	Set non-zero so that gateway pinger uses TCP pings rather than **ICMP** pings.
Common.Bigip.Bigd.HostLookup = 0	Set to non-zero to allow **bigd** to do host lookups.
Common.Bigip.Bigd.DbgFile = "/var/log/bigdlog.<bigd_pid>"	Open a debug output (log) file for **bigd**.

Table 9.4 *The BIG/db keys that store **bigd** information*

Other keys

Table 9.5 lists other keys contained in BIG/dba, including the Akamai proxy key for the content converter, the Realserver, WMI, and SNMPDCA agent keys for dynamic ration load balancing, and the CORBA keys.

Key Names	Description
Common.Bigip.Proxy.AkamaiConfigFile=/etc/config/akamai.conf	The default configuration file to use with a proxy when akamaization is enabled.
Common.Bigip.HttpAgents.WMI.LogEnabled = "true" Common.Bigip.HttpAgents.RealServer.LogEnabled = "true" Common.Bigip.SNMPDCA.LogEnabled = "true"	Open a debug output file for each of the respective monitors # when set to "true" or "yes"
Common.Bigip.CORBA.IIOPPort ="683"	Default CORBA IIOP port used for LINK-IT BIG/api
Common.Bigip.CORBA.SSLPort ="684"	Default CORBA IIOP SSL port used for LINK-IT BIG/api

Table 9.5 *Other BIG/db keys*

Key Names	Description
Common.Bigip.CORBA.AddrResolveNumeric="true"	Set to "true" causes the CORBA portal to resolve client addresses numerically
Common.Bigip.CORBA.IIOPPort ="683"	Default CORBA IIOP port used for LINK-IT BIG/api

Table 9.5 *Other BIG/db keys*

10

Configuration Files

- BIG-IP configuration files

BIG-IP configuration files

The following table includes a list of the configuration files on the BIG-IP.

File	Description
/config/bigip.conf	Stores virtual server and node definitions and settings, including node ping settings, the load balancing mode, and NAT and SNAT settings.
/config/bigip_base.conf	Stores BIG-IP self IP addresses and VLAN and interface configurations.
/config/bigip.license	Stores authorization information for the BIG-IP.
/etc/bigconf.conf	Stores the user preferences for the Configuration utility.
/config/bigconfig/openssl.conf	This file holds the configuration information for how the SSL library interacts with browsers, and how key information is generated.
/config/user.db	This is the location of the BIG/db database. This database holds various configuration information.
/config/bigconfig/httpd.conf	The main configuration file for the webserver.
/config/bigconfig/users	The webserver password file. Contains the user names and passwords of the people permitted to access whatever is provided by the webserver.
/etc/hosts	Stores the hosts table for the BIG-IP.
/etc/hosts.allow	Stores the IP addresses of workstations that are allowed to make administrative shell connections to the BIG-IP.
/etc/hosts.deny	Stores the IP addresses of workstations that are allowed to make administrative shell connections to the BIG-IP.
/etc/ipfw.conf	Stores IP filter settings.
/etc/rateclass.conf	Stores rate class definitions.
/etc/ipfwrate.conf	Stores IP filter settings for filters that also use rate classes.
/etc/snmpd.conf	Stores SNMP configuration settings.
/etc/snmptrap.conf	Stores SNMP trap configuration settings.
/config/ssh	Contains the SSH configuration and key files.
/etc/sshd_config	This is the configuration file for the secure shell server (SSH). It contains all the access information for people trying to get into the system by using SSH.
/config/routes	Contains static route information.

11

Monitoring and Administration

- Monitoring and administration utilities
- Using the bigpipe utility as a monitoring tool
- Using the Configuration utility for administration and monitoring
- Working with the BIG/top utility
- Working with the Syslog utility
- Powering down the BIG-IP
- Removing and returning items to service
- Viewing system statistics and log files
- Printing the connection table
- Changing passwords
- Working with the BIG/db database
- Working with the BIG/stat utility

Monitoring and administration utilities

The BIG-IP platform provides several utilities for monitoring and administration of the BIG-IP. You can monitor system statistics, as well as statistics specific to virtual servers and nodes, such as the number of current connections, and the number of packets processed since the last reboot.

The BIG-IP platform provides the following monitoring and configuration and administration utilities:

- **bigpipe**
  If you type certain **bigpipe** commands, such as **bigpipe virtual** or **bigpipe node,** and use the **show** keyword in the command, the command displays statistical information about the elements that you configure using that command. You can also use **bigpipe** commands to selectively reset any statistic collected by the BIG-IP.
- **The Configuration utility**
  You can use the Configuration utility to configure any feature on the BIG-IP. You can reset any statistic, or all statistics, for virtual servers, nodes, NATs, and SNATs in the Configuration utility.
- **BIG/stat**
  This utility is provided specifically for statistical monitoring of virtual servers, nodes, NATs, SNATs, and services. One benefit of using BIG/stat is that you customize the display of statistical information.
- **BIG/top**
  BIG/top provides statistical monitoring. You can set a refresh interval, and you can specify a sort order.
- **Syslog**
  Syslog is the standard UNIX system logging utility, which monitors critical system events, as well as configuration changes made on the BIG-IP.
- **BIG/db**
  BIG/db is a database that contains various configuration information for the BIG-IP.

Using the bigpipe utility as a monitoring tool

Using the **bigpipe** utility, you can view information about the BIG-IP itself, as well as elements such as virtual servers, virtual addresses, virtual ports, nodes, and node addresses. Typically, the **bigpipe** utility provides the following statistics:

- Current number of connections
- Maximum number of concurrent connections
- Total number of connections since the last system reboot
- Total number of bits (inbound, outbound, total)
- Total number of packets (inbound, outbound, total)

Monitoring the BIG-IP

The **bigpipe summary** command displays performance statistics for the BIG-IP itself. This display summary includes current usage statistics, such as the amount of time a BIG-IP has been running since the last reboot. To display a summary of the performance statistics for the BIG-IP, type the following command:

```
b summary
```

The performance statistics display in the format shown in Figure 11.1 (the output has been truncated for this example).

```
BIG-IP total uptime           = 1 (day) 4 (hr) 40 (min) 8 (sec)
BIG-IP total uptime (secs)    = 103208
BIG-IP total # connections    = 0
BIG-IP total # pkts           = 0
BIG-IP total # bits           = 0
BIG-IP total # pkts(inbound)  = 0
BIG-IP total # bits(inbound)  = 0
BIG-IP total # pkts(outbound) = 0
BIG-IP total # bits(outbound) = 0
BIG-IP error no nodes available          = 0
BIG-IP tcp port deny                     = 0
BIG-IP udp port deny                     = 0
BIG-IP virtual tcp port deny             = 0
BIG-IP virtual udp port deny             = 0
BIG-IP max connections deny              = 0
BIG-IP virtual duplicate syn ssl         = 0
BIG-IP virtual duplicate syn wrong dest  = 0
BIG-IP virtual duplicate syn node down   = 0
BIG-IP virtual maint mode deny           = 0
BIG-IP virtual addr max connections deny = 0
BIG-IP virtual path max connections deny = 0
BIG-IP virtual non syn                   = 0
BIG-IP error not in out table            = 0
BIG-IP error not in in table             = 0
BIG-IP error virtual fragment no port    = 0
BIG-IP error virtual fragment no conn    = 0
BIG-IP error standby shared drop         = 0
BIG-IP dropped inbound                   = 0
BIG-IP dropped outbound                  = 0
BIG-IP reaped                            = 0
BIG-IP ssl reaped                        = 0
BIG-IP persist reaped                    = 0
BIG-IP udp reaped                        = 0
BIG-IP malloc errors                     = 0
BIG-IP bad type                          = 0
BIG-IP mem pool total 96636758 mem pool used 95552 mem percent used   0.10
```

Figure 11.1 *The **bigpipe** summary display screen*

Table 11.1 contains descriptions of each individual statistic included in the summary display screen.

Statistic	Description
total uptime	Total time elapsed since the BIG-IP was last booted.
total uptime (secs)	Total uptime displayed in seconds.
total # connections	Total number of connections handled.
total # pkts	Total number of packets handled.
total # bits	Total number of bits handled.
total # pkts (inbound)	Total number of incoming packets handled.
total # bits (inbound)	Total number of incoming bits handled.
total # pkts (outbound)	Total number of outgoing packets handled.
total # bits (outbound)	Total number of outgoing bits handled.
error no nodes available	The number of times the BIG-IP tried to make a connection to a node, but no nodes were available.
tcp port deny	The number of times a client attempted to connect to an unauthorized TCP port on the BIG-IP (unauthorized port and source IP are logged in the **syslog**).
udp port deny	The number of times a client attempted to connect to an unauthorized UDP port on the BIG-IP (unauthorized port and source IP are logged in the **syslog**).
virtual tcp port deny	The number of times a client attempted to connect to an unauthorized TCP port on a virtual address (unauthorized port and source IP are logged in the **syslog**).
virtual udp port deny	The number of times a client attempted to connect to an unauthorized UDP port on a virtual address (unauthorized port and source IP are logged in the **syslog**).
max connections deny	The total number of connections denied because the maximum number of connections allowed was exceeded.
virtual duplicate syn ssl	The number of duplicate connection attempts to existing SSL connections from the same client.
virtual duplicate syn wrong dest	The number of duplicate connection attempts from the same client (address and port combination) to a different virtual server.
virtual duplicate syn node down	The number of duplicate connection attempts to a server that is down when a connection to the server was made previously.
virtual maint mode deny	The number of times a connection to a virtual server was denied while the BIG-IP is in maintenance mode.
virtual addr max connections deny	The number of virtual address connections dropped because the maximum number of connections was exceeded.

***Table 11.1 bigpipe** monitoring statistics*

Statistic	Description
virtual path max connections deny	The number of virtual path connections dropped because the maximum number of connections was exceeded.
virtual non syn	The number of packets received which are not connection requests, and are destined to a virtual address, but not a valid virtual server (port).
error virtual fragment no port	The number of IP fragments for which there is no port.
error virtual fragment no conn	The number of IP fragments for which there is no connection.
error standby shared drop	The number of packets destined to the shared IP address in a redundant system that are received and ignored by the standby system.
dropped inbound	The total number of inbound packets dropped by the BIG-IP.
dropped outbound	The total number of outbound packets dropped by the BIG-IP.
reaped	The total number of connections that timed-out, and are deleted by the BIG-IP.
ssl reaped	The total number of SSL session ID records that timed-out, and were closed by the BIG-IP.
persist reaped	The total number of persistence records that timed-out, and were closed by the BIG-IP.
udp reaped	The total number of UDP connections that timed-out, and were closed by the BIG-IP.
malloc errors	The number of times a connection could not be created because the system is low on memory.
mem pool total	The total amount of memory available in all combined memory pools.
mem pool used	The total amount of memory, in all combined memory pools, in use by the BIG-IP.
mem percent used	The total percentage of memory in use by all combined memory pools.

***Table 11.1 bigpipe** monitoring statistics*

Resetting statistics on the BIG-IP

The **bigpipe** commands allow you to selectively reset any statistic on the BIG-IP. The statistics you can reset selectively include:

- Virtual address
- Virtual server
- Node address
- Node server
- Virtual port
- Network address translations (NATs)

- Secure network address translations (SNATs)
- Global statistics

When you reset one of these items, the packets in, packets out, bytes in, and bytes out counters of the target item are reset to zero. The maximum connection count counter is also reset. The current connections counter is not reset, and the total connections counter is set equal to the number of current connections.

◆ Note

The statistics are reset for the specified items only. Statistics for dependent items, such as node servers for a given virtual address, are not modified by these commands. The only exception is the global statistics reset option which resets traffic statistics for all items. After an item-level reset, statistics for all other dependent items do not add up.

You can create an audit trail for reset events by setting an optional system control variable. You can set this variable to generate a syslog log entry. To set this variable, type the following command:

```
b internal set verbose_log_level=4
```

To reset statistics for virtual servers and virtual addresses

Use the following syntax to reset statistics for the virtual address specified by the IP address **<virtual ip>**.

```
b virtual <virtual_ip> stats reset
```

For example, if you want to reset statistics for the virtual address **172.20.1.100**, type the following command:

```
b virtual 172.20.1.100 stats reset
```

If you want to reset statistics for a list of virtual addresses, type the command with a list of addresses separated by spaces:

```
b virtual 172.20.1.100 172.20.1.101 172.20.1.102 stats reset
```

If you want to reset statistics for all virtual servers, use the following command:

```
b virtual stats reset
```

Use the following syntax to reset statistics for the virtual server IP:port combination **<virtual_ip>:<port>**.

```
b virtual <virtual_ip>:<port> stats reset
```

For example, if you want to reset statistics for the virtual address/port combination **172.20.1.100:80**, type the following command:

```
b virtual 172.20.1.100:80 stats reset
```

If you want to reset statistics for a list of virtual address/port combinations, type the command with the list of addresses separated by spaces:

```
b virtual 172.20.1.100:80 172.20.1.100:23 172.20.1.101:80 stats reset
```

To reset statistics for node servers and node addresses

Use the following syntax to reset statistics for all node addresses and node servers:

```
b node stats reset
```

You can reset statistics for the node address specified by the IP address <**node_ip**>:

```
b node <node_ip> stats reset
```

For example, to reset the statistics for the node address 10.1.1.1, use the following syntax:

```
b node 10.1.1.1 stats reset
```

If you want to reset statistics for a list of node addresses, type the command with the list of addresses separated by spaces:

```
b node 10.1.1.1 10.1.1.2 10.1.1.3 stats reset
```

Use the following syntax to reset statistics for the node server specified by the IP:port combination <**node_ip**>:<**port**>:

```
b node <node_ip>:<port> stats reset
```

For example, to reset the statistics for the node server **10.1.1.1:80**, use the following syntax:

```
b node 10.1.1.1:80 stats reset
```

If you want to reset statistics for a list of node server addresses, type the command with the list of addresses separated by spaces:

```
b node 10.1.1.1:80 10.1.1.2:23 stats reset
```

To reset statistics for virtual ports

Use the following syntax to reset statistics for all virtual ports:

```
b service stats reset
```

Use the following syntax to reset statistics for the virtual port <**port**>. You can specify a list of virtual ports separated by spaces:

```
b service <port> stats reset
```

For example, to reset the statistics for the virtual port **80**, use the following command:

```
b service 80 stats reset
```

To reset the statistics for a list of virtual ports, use the following syntax:

```
b service 23 80 443 stats reset
```

To reset statistics for network address translations (NATs)

Use the following syntax to reset statistics for all NATs:

```
b nat stats reset
```

Use the following syntax to reset statistics for the NAT for the IP address <**orig_ip**>.

```
b nat <orig_ip> stats reset
```

For example, to reset the statistics for the NAT **172.20.3.101**, use the following command:

```
b nat 172.20.3.101 stats reset
```

To reset the statistics for a list of origin IPs, use the following command where addresses are separated by spaces:

```
b nat 172.20.3.101 172.20.3.102 stats reset
```

To reset statistics for secure network address translations (SNATs)

Use the following syntax to reset statistics for all SNATs:

```
b snat stats reset
```

Use the following syntax to reset statistics for the SNAT for IP address <**snat_ip**>:

```
b snat <snat_ip> stats reset
```

For example, to reset the statistics for the SNAT **172.20.3.101**, use the following command:

```
b snat 172.20.3.101 stats reset
```

To reset the statistics for a list of SNAT origin addresses, use the following command where addresses are separated by spaces:

```
b snat 172.20.3.101 172.20.3.102 stats reset
```

To reset global statistics

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

```
b global stats reset
```

To reset any statistic in the Configuration utility

A **Reset** button is located in the Configuration utility in each of the following tables:

- Virtual address
- Virtual server
- Node address
- Node server
- Virtual port
- Network address translations (NATs)
- Global statistics

To reset a statistic for a particular item, click the **Reset** button next to the item in one of these tables.

Monitoring virtual servers, virtual addresses and services

You can use different variations of the **bigpipe virtual** command, as well as the **bigpipe port** command, to monitor information about virtual servers, virtual addresses, and services managed by the BIG-IP.

Displaying information about virtual servers and virtual addresses

The **bigpipe virtual** command displays the status of virtual servers (**up**, **down**, **unchecked**, or **disabled**), the current number of connections to each virtual server, and the status of the member nodes that are included in each virtual server mapping. The status for individual member nodes includes whether the node is **up**, **down**, **unchecked**, or **disabled** and also includes the cumulative count of packets and bits received and sent by the node on behalf of the virtual server. The BIG-IP displays the statistics as shown in Figure 11.2.

```
virtual +------> 11.11.11.50         UNIT 1
    |            (cur, max, limit, tot) = (0, 8, 0, 370)
    |            (pckts,bits) in = (10704, 8744872), out = (21480, 230874016)
    +---+--> PORT http                 UP
        |        (cur, max, limit, tot) = (0, 8, 0, 370)
        |        (pckts,bits) in = (10704, 8744872), out = (21480, 230874016)
        POOL appgen_11.11.11.50.80
        MEMBER 11.12.11.100:http       UP
                 (cur, max, limit, tot) = (0, 8, 0, 370)
                 (pckts,bits) in = (10704, 8744872), out = (21480, 230874016)

virtual +------> 11.11.11.101        UNIT 1
    |            (cur, max, limit, tot) = (0, 2, 0, 4)
    |            (pckts,bits) in = (4532, 2090768), out = (6824, 82113984)
    +---+--> PORT http                 UP
        |        (cur, max, limit, tot) = (0, 2, 0, 4)
        |        (pckts,bits) in = (4532, 2090768), out = (6824, 82113984)
        POOL my_website_pool
        MEMBER 11.12.11.100:http       UP
                 (cur, max, limit, tot) = (0, 2, 0, 4)
                 (pckts,bits) in = (4532, 2090768), out = (6824, 82113984)
```

Figure 11.2 *Virtual server statistics*

If you want to view statistical information about one or more specific virtual servers, simply include the virtual servers in the **bigpipe virtual show** command as shown below:

```
b virtual <virt addr>:<port> ... <virt addr>:<port> show
```

If you want to view statistical information about traffic going to one or more virtual addresses, specify only the virtual address information in the command:

```
b virtual <virt addr> ... <virt addr> show
```

Displaying information about services

The **bigpipe port show** command allows you to display information about specific virtual ports managed by the BIG-IP. You can use the command to display information about all virtual services, or you can specify one or more particular virtual services.

To view information about all virtual services, use the following syntax:

```
b service show
```

To view statistical information about one or more specific virtual services, simply include the service names or port numbers as shown below:

```
b service <port> ... <port> show
```

Monitoring nodes and node addresses

The **bigpipe node** command displays the status of all nodes configured on the BIG-IP. The information includes whether the specified node is **up**, **down**, **disabled**, or **unchecked**, and the number of cumulative packets and bits sent and received by each node on behalf of all virtual servers. The BIG-IP displays the statistical information as shown in Figure 11.3.

```
NODE 11.12.11.100      UP
|        (cur, max, limit, tot) = (0, 8, 0, 374)
|        (pckts,bits) in = (15236, 10835640), out = (28304, 312988000)
+-     PORT http       UP
         (cur, max, limit, tot) = (0, 8, 0, 374)
         (pckts,bits) in = (15236, 10835640), out = (28304, 312988000)
```

Figure 11.3 *Node statistics screen*

If you want to view statistical information about one or more specific nodes, simply include the nodes in the **bigpipe node show** command as shown below:

```
b node <node addr>:<port> ... <node addr>:<port> show
```

If you want to view statistical information about traffic going to one or more node addresses, specify only the node address information in the command:

```
b node <node addr> ... <node addr> show
```

Monitoring NATs

The **bigpipe nat show** command displays the status of the NATs configured on the BIG-IP. The information includes the number of cumulative packets and bits sent and received by each NAT.

To display NAT status from the command line

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

```
b nat show
```

Use the following syntax to display the status of one or more selected NATs:

```
b nat <node addr> [...<node addr>] show
```

An example of the output for this command is shown in Figure 11.4.

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

Figure 11.4 *NAT statistics*

Monitoring SNATs

The **bigpipe snat show** command displays the status of the SNATs configured on the BIG-IP. The information includes connections and global SNAT settings.

To show SNAT details from the command line

Use the following **bigpipe** command to show SNAT mappings:

```
b snat [<SNAT addr>] [...<SNAT addr>] show
b snat show
```

Use the following command to show the current SNAT connections:

```
b snat [<snat_ip>...] dump [ verbose ]
b snat dump [ verbose ]
```

The optional **verbose** keyword provides more detailed output.

The following command prints the global SNAT settings:

```
b snat globals show
```

Viewing the status of the interface cards

The **bigpipe interface** command displays the current status and the settings for external and internal interface cards. You can also use the **bigpipe interface** command to view information for a specific interface card, using the following command syntax:

```
b interface <ifname> -show
```

Using the Configuration utility for administration and monitoring

You can use the Configuration utility System Admin screen to add users, customize the user interface, configure SNMP, and save and restore a current configuration.

You can use the Configuration utility to allow access to the SNMP agent and to set SNMP properties. For more information on configuring SNMP, refer to Chapter *8, Configuring SNMP*.

Adding a user

You can add a user to the BIG-IP using the Configuration utility.

To add a user to the BIG-IP using the Configuration utility

1. In the navigation pane, click **System Admin**.
   The System Admin tabs appear.
2. Click the User Administration tab.
   The Add User screen opens. This screen contains a list of current users.
3. In the appropriate field, type the User ID, password, and access level for the user.
   For more information on the Add User screen, click the **Help** button.

Customizing the Configuration utility

You can customize the appearance of the Configuration utility.

To customize the Configuration utility

1. In the navigation pane, click **System Admin**.
   The System Admin tabs appear.
2. Click the Web UI Administration tab.
   The WEB UI Administration screen opens.
3. Select the options you want to configure.
   For more information about the options available on this screen, click the **Help** button.

Configuring SNMP

For information on configuring SNMP, refer to Chapter *8, Configuring SNMP*.

Working with the BIG/top utility

BIG/top™ is a real-time statistics display utility. The display shows the date and time of the latest reboot and lists activity in bits, bytes, or packets. Similar to BIG/stat, the BIG/top utility accepts options which allow you to customize the display of information. For example, you can set the interval at which the data is refreshed, and you can specify a sort order. The BIG/top displays the statistics as shown in Figure 11.5, following.

```
                   |  bits  since       |  bits in prior     |   current
                   |  Nov 28 18:47:50   |  3 seconds         |   time
BIG-IP      ACTIVE |---In----Out---Conn-|---In----Out---Conn-|   00:31:59
227.19.162.82          1.1G  29.6G 145     1.6K     0      0

virtual ip:port        |---In----Out---Conn-|---In----Out---Conn-|-Nodes Up--
217.87.185.5:80       1.0G  27.4G 139.6K   1.6K     0      0       2
217.87.185.5:20      47.5M  2.1G    3.1K     0      0      0       2
217.87.185.5:20      10.2M 11.5M    2.6K     0      0      0       2

NODE  ip:port      |---In----Out---Conn-|---In----Out---Conn-|--State----
129.186.40.17:80    960.6M  27.4G  69.8K   672      0      0      UP
129.186.40.17:20     47.4M   2.1G   3.1K     0      0      0      UP
129.186.40.18:80    105.3M 189.0K  69.8K   1.0K     0      0      UP
129.186.40.17.21      9.4M  11.1M   1.3K     0      0      0      UP
129.186.40.18:21    700.8K 414.7K   1.3K     0      0      0      UP
129.186.40.18:20      352    320      1      0      0      0      UP
```

Figure 11.5 *The BIG/top screen display*

Using BIG/top command options

The **bigtop** command supports the options and syntax outlined in Table 11.2:

```
bigtop  [options...]
```

Option	Description
-bytes	Displays counts in bytes (the default is bits).
-conn	Sorts by connection count (the default is to sort by byte count).
-delay <value>	Sets the interval at which data is refreshed (the default is four seconds).
-delta	Sorts by count since last sample (the default is to sort by total count).
-help	Displays BIG/top help.
-nodes <value>	Sets the number of nodes to print (the default is to print all nodes).
-nosort	Disables sorting.
-once	Prints the information once and exits.

Table 11.2 *BIG/top command options*

Option	Description
-pkts	Displays the counts in packets (the default is bits).
-scroll	Disables full-screen mode.
-virtuals <value>	Sets the number of virtual servers to print (the default is to print all virtual servers).

Table 11.2 *BIG/top command options*

Using runtime commands in BIG/top

Unless you specified the **-once** option, the BIG/top utility continually updates the display at the rate indicated by the **-delay** option. You can also use the following runtime options at any time:

- The **u** option cycles through the display modes: bits, bytes, and packets.
- The **q** option quits the BIG/top utility.

Working with the Syslog utility

The BIG-IP supports logging using the **Syslog** utility. The logs are generated automatically, and saved in user-specified files. These logs contain all changes made to the BIG-IP configuration, such as those made with the **bigpipe virtual** command, or other **bigpipe** commands, as well as all critical events that occur in the system.

◆ Note

You can configure the Syslog utility to send email or activate pager notification based on the priority of the logged event.

The Syslog log files track system events based on information defined in the **/etc/syslog.conf** file. You can view the log files in a standard text editor, or with the **less** file page utility.

Sample log messages

Table 11.3 shows sample log messages to give you an idea of how the Syslog utility tracks events that are specific to the BIG-IP.

Sample message	Description
bigd: allowing connections on port 20	A user specifically allowed connections on virtual port 20.
bigd: node 192.168.1.1 detected up	The 192.168.1.1 node address was successfully pinged by the BIG-IP.
bigd: added service port 20 to node 192.168.1.1	A user defined a new node, 192.168.1.1:20.
kernel: security: port denial 207.17.112.254:4379 -> 192.168.1.1:23	A client was denied access to a specific port. The client is identified as coming from 207.17.112.254:4379, and the destination node is 192.168.1.1:23.

Table 11.3 *Sample Syslog messages*

Powering down the BIG-IP

If you want to power down, or turn off, the BIG-IP you need to complete two tasks. The first task is to shut down the BIG-IP software. After you shut down the BIG-IP software, you can turn the power to the system off.

To shut down the BIG-IP software from the command line

To complete the first task to shut down the BIG-IP software, type the following command:

```
halt
```

After the system halts, you can turn the power to the system off.

Removing and returning items to service

Once you have completed the initial configuration on the BIG-IP, you may want to temporarily remove specific items from service for maintenance purposes. For example, if a specific network server needs to be upgraded, you may want to disable the nodes associated with that server, and then enable them once you finish installing the new hardware and bring the server back online.

If you specifically disable the nodes associated with the server, the BIG-IP allows the node to go down only after all the current connections are complete. During this time, the BIG-IP does not attempt to send new connections to the node. Although the BIG-IP monitoring features would

eventually determine that the nodes associated with the server are down, specifically removing the nodes from service can prevent interruptions on long duration client connections.

You can remove the entire BIG-IP from service, or you can remove the following individual items from service:

- Virtual servers
- Virtual addresses
- Virtual ports
- Nodes
- Node addresses

Removing the BIG-IP from service

The BIG-IP platform offers a Maintenance mode, which allows you to remove the BIG-IP from network service. This is useful if you want to perform hardware maintenance, or make extensive configuration changes. When you activate Maintenance mode, the BIG-IP no longer accepts connections to the virtual servers it manages. However, the existing connections are allowed to finish processing so that current clients are not interrupted.

The **bigpipe maint** command toggles the BIG-IP into or out of Maintenance mode. Use the following command to put the BIG-IP in maintenance mode:

```
b maint
```

If the BIG-IP runs in Maintenance mode for less than 20 minutes and you return the machine to the normal service, the BIG-IP quickly begins accepting connections. However, if the BIG-IP runs in Maintenance mode for more than 20 minutes, returning the unit to service involves updating all network ARP caches. This process can take a few seconds, but you can speed the process up by reloading the **/config/bigip.conf** file using the following command:

```
b -f /config/bigip.conf
```

To activate maintenance mode using the Configuration utility

1. In the navigation pane, click **System**.
   The Network Map screen opens.
2. Click the Properties tab.
   The Properties screen opens.
3. Check the **Maintenance Mode** box.
4. Click the **Apply** button.

Removing individual virtual servers, virtual addresses, and ports from service

The BIG-IP also supports taking only selected virtual servers, addresses, or ports out of service, rather than removing the BIG-IP itself from service. Each **bigpipe** command that defines virtual servers and their components supports **enable** and **disable** keywords, which allow you to remove or return the elements from service.

When you remove a virtual address or a virtual port from service, it affects all virtual servers associated with the virtual address or virtual port. Similarly, if you remove a node address from service, it affects all nodes associated with the node address.

Enabling and disabling virtual servers and virtual addresses

The **bigpipe virtual** command allows you to enable or disable individual virtual servers, as well as virtual addresses.

To enable or disable a virtual server from the command line

To enable or disable a virtual server, type the appropriate command:

```
b virtual <virtual addr>:<virtual port> enable
b virtual <virtual addr>:<virtual port> disable
```

To enable or disable a virtual address, type the appropriate command:

```
b virtual <virtual addr> enable
b virtual <virtual addr> disable
```

Enabling and disabling virtual ports

The **bigpipe port** command allows you to allow or deny traffic on a virtual port.

To allow or deny traffic on a virtual port from the command line

Use the following syntax to allow or deny traffic on a virtual port.

```
b service <virtual port> enable
b service <virtual port> disable
```

Removing individual nodes and node addresses from service

You can enable or disable individual and node addresses from the command line.

To enable and disable nodes and node addresses from the command line

The **bigpipe node** command allows you to enable or disable individual nodes, as well as node addresses.

To enable or disable a **node**, type the appropriate command:

```
b node <node addr>:<node port> enable
b node <node addr>:<node port> disable
```

To enable or disable a **node address**, type the appropriate command:

```
b node <node addr> enable
b node <node addr> disable
```

Viewing the currently defined virtual servers and nodes

When used with the **show** parameter, **bigpipe** commands typically display currently configured elements. For example, the **bigpipe virtual show** command displays all currently defined virtual servers, and the **bigpipe node** command displays all nodes currently included in virtual server mappings. For additional information about using **bigpipe** commands on the BIG-IP, see Chapter 7, *bigpipe Command Reference*.

Viewing system statistics and log files

The Configuration utility allows you to view a variety of system statistics and system log files. Note that from each statistics screen, you can access property settings for individual virtual servers, nodes, IP addresses, and ports by selecting the individual item in the statistics table.

Viewing system statistics

The Configuration utility allows you to view the following statistical information:

- BIG-IP system statistics, including the elapsed time since the last system reboot, the number of packets and connections handled by the system, and the number of dropped connections
- Virtual servers, including virtual servers, virtual address only, or virtual ports only
- Nodes, including nodes, node addresses only, or node ports only
- NAT statistics, such as the number of packets handled by each NAT
- SNAT statistics, such as SNAT mappings
- IP filter statistics, including the number of packets accepted and rejected by individual IP filters

- Rate filter statistics, including the number of bits passed through, delayed, and dropped by individual rate filters
- Information about illegal connection attempts, such as the source IP addresses from which the illegal connection is initiated

Statistics are displayed in real-time. You can specify the update frequency by setting an interval (in seconds), and then clicking **Update**.

Viewing log files

The Configuration utility allows you to display three different log files:

- The BIG-IP system log, which displays standard UNIX system events
- The BIG-IP log, which displays information specific to BIG-IP events, such as defining a virtual server
- The Pinger log, which displays status information determined by each node ping issued by the BIG-IP

Printing the connection table

The **bigpipe** command line utility also offers a useful diagnostic tool that prints the list of current connections. Normally, the **bigpipe conn** command prints the client, virtual server, and node addresses.

Changing passwords

When you run the Setup utility, you define a password that allows remote access to the BIG-IP, and you also define a password for the BIG-IP web server. You can change these passwords at any time.

To change the BIG-IP password

1. At the BIG-IP command line prompt, log on as the root user and use the **passwd** command.
2. At the password prompt, enter the password you want to use for the BIG-IP and press Enter.
3. To confirm the password, retype it and press Enter.

Changing passwords and adding new user IDs for the web-based Configuration utility

You can create new users for the BIG-IP web server in the Configuration utility.

The user accounts you create in the Configuration utility can have full, partial, or read-only access to the BIG-IP.

To create user accounts in the Configuration utility

1. In the navigation pane, click **User Admin**.
   The User Administration screen opens.
2. In the Add User section, type the following information.
   - **User ID**
     Type the user ID you want to assign the user.
   - **Password**
     Type the password you want to assign the user.
   - **Retype Password**
     Retype the password you want to assign the user.
3. In the Current Users list, select the access level for the user. The access levels available are:
   - **Read Only**
     This access level allows the user only to view information in the Configuration utility. Users with this access level do not have access to **Add** buttons, certain tab items, **Apply** buttons, or **Remove** buttons.
   - **Partial Read/Write**
     In addition to allowing the user to view information, a Partial Read/Write user can also change the status of node addresses to either **enabled** or **disabled**.
   - **Full Read/Write**
     This access level provides the user with full access to all administrative tasks.
4. After you select the access level for the user, click the **Add** button.

The Current User list on the User Administration screen contains all users configured to access the Configuration utility. You can delete any user added through the Configuration utility by clicking the **Remove** button next to the user in the list. The BIG-IP web server administrator account you created with the Setup utility shows up in this list. However, you cannot edit or delete this account from the Configuration utility. To edit this account, you must run the **config httpd** command line utility. For more information about this utility, see Chapter 2, *Using the Setup Utility*.

Working with the BIG/db database

The BIG/db™ database holds certain configuration information for the BIG-IP. Most BIG-IP utilities currently use the configuration stored in BIG/db. The **bigpipe db** is provided for loading configuration information into BIG/db. An additional **default.txt** file is included with the BIG-IP which contains default information you can load into the BIG/db database.

Using the bigpipe db command

The keys are viewed and set using the bigpipe **db** command.

```
b db get <key>
b db get <reg_exp>
b db set <key>
b db set <key> = <value>
b db unset <key>
b db unset <reg_exp>
b db dump [filename]
```

To display current setting of a BIG/db configuration key

To display the value of a BIG/db configuration key, use the following syntax:

```
b db get <key>
b db get <regular_exp>
```

For example, the following command displays the value of **Local.Bigip.FTB.HostNumber**:

```
b db get Local.Bigip.FTB.HostNumber
```

The following command displays the value of all local keys:

```
b db get Local.*
```

To set a BIG/db configuration key

To create (set) a BIG/db configuration key, use the following syntax:

```
b db set <key>
```

To set a BIG/db configuration key and assign a value to it, use the following syntax

```
b db set <key> = <value>
```

For example, the following command sets **Local.Bigip.FTB.HostNumber** mode to **on**:

```
b db set Local.Bigip.FTB.HostNumber = 1
```

To unset a BIG/db configuration key

To unset the a BIG/db configuration key, use the following syntax.

```
b db unset <key>
b db unset <regular_exp>
```

For example, the following command unsets **Local.Bigip.FTB.HostNumber**:

```
b db unset Local.Bigip.FTB.HostNumber
```

The following command unsets all local keys:

```
b db unset set Local.*
```

Working with the default.txt file

The **default.txt** file documents the keys that are valid in the BIG/store database. This file is located at **/config/default.txt**. It contains all the possible database keys, comments that document these keys, and the default values used by programs that run on the BIG-IP.

◆ Note

*The values in the **default.txt** file are default values, several of the keys listed are not present in the BIG/db database.*

The **default.txt** file is intended to serve as documentation only. Some of the records, such as those that represent IP addresses and port numbers, need to be set to values other than the default values for the system to work. Additionally, some of the key names listed are wildcard keys. These keys are not valid key names.

If you want to load **default.txt** into the BIG/db database, we recommend that you dump the existing database to another text file. Make a copy of **default.txt**, and then edit the copy so that the records which are present in your dump file match the values contained in the **default.txt file**. After the values match, you can load the edited copy of **default.txt**.

For a complete list of the keys available in the BIG/db, see Chapter 9, *BIG/db Configuration Keys*.

Working with the BIG/stat utility

BIG/stat™ is a utility that allows you to quickly view the status of the following elements:

- Virtual servers
- Services (**cur**, **max**, **limit**, **tot**) (**pckts**,**bits**) in out
- Nodes (**cur**, **max**, **limit**, **tot**) (**pckts**,**bits**) in out
- Ports
- Network address translations (NATs)

You can customize the BIG/stat utility statistics display. For example, you can customize your output to display statistics for a single element, or for selected elements. You can set the display to automatically update at time intervals you specify.

The **bigstat** command accepts one or more options, which allow you to customize the statistical display. When you use the **bigstat** command without specifying any options, the BIG/stat utility displays the list of virtual servers, services, nodes, NATs, and SNATs only one time. The basic command syntax is:

```
bigstat [ options...]
```

The following table, Table 11.4, describes the options that you can use in the **bigstat** command.

Option	Description
-bigip	Displays totals for the BIG-IP overall.
-c <count>	Sets the interval at which new information is displayed.
-h and -help	Displays the help options.
-n	Displays data in numeric format.
-nat	Displays network address table (NAT) entries only.
-no_virtualtot	Removes virtual server totals from the display.
-no_nodetot	Removes node totals from the display.
-node	Displays nodes only.
-port	Displays ports only.
-v	Displays version information.
-virtual	Displays virtual servers only.

Table 11.4 *The **bigstat** command options*

Figure 11.6 contains an example of the output from the **bigstat** command. Table 11.5 contains descriptions of each of the items in this example.

```
bigip springbank
         (cur, max, tot) = (0, 8, 374)
         (pckts,bits) in = (15310, 10860064),  out = (28363, 313009048)
virtual   11.11.11.50
         (cur, max, limit, tot) = (0, 8, 370, 370)
         (pckts,bits) in = (10704, 8744872), out = (21480, 230874016)
virtual   11.11.11.50:http        UP
         (cur, max, limit, tot) = (0, 8, 370, 370)
         (pckts,bits) in = (10704, 8744872), out = (21480, 230874016)
virtual   11.11.11.101
         (cur, max, limit, tot) = (0, 2, 4, 4)
         (pckts,bits) in = (4532, 2090768), out = (6824, 82113984)
virtual   11.11.11.101:http       UP
         (cur, max, limit, tot) = (0, 2, 4, 4)
         (pckts,bits) in = (4532, 2090768), out = (6824, 82113984)
node  11.12.11.100            UP
         (cur, max, limit, tot) = (0, 8, 374, 374)
         (pckts,bits) in = (15236, 10835640), out = (28304, 312988000)
node  11.12.11.100:http       UP
         (cur, max, limit, tot) = (0, 8, 374, 374)
         (pckts,bits) in = (15236, 10835640), out = (28304, 312988000)
port  WILDCARD PORT
         (cur, max, limit, tot, reaped) = (0, 0, 0, 0, 0)
         (pckts,bits) in = (0, 0), out = (0, 0)
port  80:http
         (cur, max, limit, tot, reaped) = (0, 8, 374, 374, 6)
         (pckts,bits) in = (15236, 10835640), out = (28304, 312988000)
```

Figure 11.6 *Sample output of the* ***bigstat*** *command*

The following table contains descriptions of each of the metrics collected for the BIG-IP.

BIG/stat Item	Description
BIG-IP	**cur -** Shows the number of current connections handled by the BIG-IP **max -** Shows the maximum number of connections handled by the BIG-IP **tot** - Shows the total number of connections handled by the BIG-IP **pckts,bits in -** Shows the total number of packets and bits coming into the BIG-IP **pckts,bits out -** Shows the total number of packets and bits going out of the BIG-IP
virtual server	**cur -** Shows the number of current connections handled by the virtual server **max -** Shows the maximum number of connections handled by the virtual server **limit** - Shows the connection limit reached by the virtual server **tot** - Shows the total number of connections handled by the virtual server **pckts,bits in -** Shows the total number of packets and bits coming into the virtual server **pckts,bits out -** Shows the total number of packets and bits going out of the virtual server
service	**cur -** Shows the number of current connections handled by the service **max -** Shows the maximum number of connections handled by the service **limit** - Shows the connection limit reached by the service **tot** - Shows the total number of connections handled by the BIG-IP service **pckts,bits in -** Shows the total number of packets and bits coming into the service **pckts,bits out -** Shows the total number of packets and bits going out of the service
nodes	**cur -** Shows the number of current connections handled by the node **max -** Shows the maximum number of connections handled by the node **limit** - Shows the connection limit reached by the node **tot** - Shows the total number of connections handled by the BIG-IP node **pckts,bits in -** Shows the total number of packets and bits coming into the node **pckts,bits out -** Shows the total number of packets and bits going out of the node
ports	**cur -** Shows the number of current connections handled by the port **max -** Shows the maximum number of connections handled by the port **limit** - Shows the connection limit reached by the port **tot** - Shows the total number of connections handled by the BIG-IP port **reaped -** Shows the number of connections reaped on the port **pckts,bits in -** Shows the total number of packets and bits coming into the port **pckts,bits out -** Shows the total number of packets and bits going out of the port

Table 11.5 *Data displayed by the* ***bigstat*** *utility*

12

Additional Setup Options

- Overview of additional setup options
- Defining additional host names
- Using the MindTerm SSH Console
- Downloading the SSH client to your administrative workstation
- Addressing general networking issues
- Using a serial terminal with the BIG-IP
- Configuring RADIUS or LDAP authentication

Overview of additional setup options

This chapter contains details about additional setup options you may want to configure for the BIG-IP. The options described in this chapter include:

- Defining additional host names
- Preparing workstations for command line access
- Addressing general networking issues
- Using a serial terminal with the BIG-IP
- Configuring RADIUS and LDAP authentication

Defining additional host names

Once you complete the Setup utility, you may want to insert additional host names and IP addresses for network devices into the **/etc/hosts** file to allow for more user-friendly system administration. In particular, you may want to create host names for the IP addresses that you will assign to virtual servers. You may also want to define host names for standard devices such as your routers, network interface cards, and the servers or other equipment that you are load balancing.

The **/etc/hosts** file, as created by the Setup utility, is similar to the example shown in Figure 12.1.

```
# BIG-IP(R) Hosts Table   Generated by Setup utility on Thu May 16 11:03:03 PDT 2002

# localhost entry
127.1   localhost

# default gateway entry
11.11.11.10     router


# Local name
11.11.11.2      bigip1.mynet.net

# Peer name (state mirror)
11.12.11.1      peer

#
# vlans
#
11.11.11.2      external
11.12.11.2      internal

#
# VIPS and NODES ( add below - do not delete this line )
#
```

Figure 12.1 *The* **/etc/hosts** *file created by the Setup utility*

This sample **hosts** file lists the IP addresses for the default router, the internal VLAN, and the external VLAN, and it contains placeholders for both the virtual servers and the content servers that the BIG-IP will manage.

*If you have modified the **/etc/hosts** file with something other than the Setup utility, such as **vi** or **pico**, be aware that your changes may be lost when you run the Setup utility (**config**). The Setup utility overwrites the **/etc/hosts** file and **openssl.conf**, but it does not warn you before doing so.*

Using the MindTerm SSH Console

With the MindTerm SSH Console, you can open an SSH session for the BIG-IP from the Configuration utility. Use the MindTerm SSH client to enable secure command line administration. You can perform any of the command line tasks in a popup console screen.

The MindTerm SSH client requires a Java virtual machine to operate. If you are unable to run the MindTerm SSH client, make sure that you have a Java virtual machine installed and that your browser has Java enabled in the Preferences, or Options, section. For more information on Java virtual machines and download options, visit your web browser manufacturer's web site.

To open the MindTerm SSH Console using the Configuration utility

1. In the navigation pane, click **MindTerm SSH Console**.
   A popup console opens.
2. When you see the command prompt, press Enter.
3. Log in to the BIG-IP as you normally would.

◆ Note

When you use the MindTerm SSH Console, you can only administer the local BIG-IP. If you wish to administer remote systems, you do so using an SSH or Telnet session from the command line. For information about installing an SSH client on the administrative workstation, see the following section.

Downloading the SSH client to your administrative workstation

From BIG-IP units that support encrypted communications, you can download the SSH client to your administrative workstation in preparation for remote command line access. In addition to running BIG-IP command line utilities, you can also use the SSH suite for file transfer to and from the BIG-IP, as well as for remote backups.

The SSH client is available for both Windows and UNIX platforms, and you can download your preferred client either from the web server or using an FTP connection. You can find detailed information about the SSH client in the documentation provided on the web server or on the Documentation and Software CD-ROM.

◆ Note

If your BIG-IP does not support encrypted connections, you can use a Telnet shell for remote command line access.

Downloading the SSH client from the web server

Connect to the BIG-IP using **https://** rather than **http://** in the URL. In the Additional Software Downloads section, click the **SSH Clients** link. From the SSH Clients page, you can choose the SSH Client appropriate to your operating system.

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

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

To unzip and install the SSH client

1. Log on to the Windows workstation.
2. Navigate to the directory to which you transferred the installation file. Run **PKZip** or **WinZip** to extract the files.
3. The set of files extracted includes a Setup program. Run the Setup program to install the client.
4. Start the SSH client.
5. In the SSH Client window, from the Edit menu choose **Properties**. The Properties dialog box opens.

6. In the Connection tab, in the Remote Host section, type the following items:
   - In the **Host Name** box, type the BIG-IP IP address or host name.
   - In the **User Name** box, type the root user name.
7. In the Options section, check **Compression** and set the Cipher option to **Blowfish**.
8. Click the **OK** button.

Setting up the SSH client on a UNIX workstation

The installation file for UNIX platforms is compressed in **tar**/**gzip** format.

To untar and install the SSH client

1. Log on to the workstation and navigate to the directory into which you transferred the SSH client tar file.
2. Untar the file and follow the instructions in the ***install*** file to build the SSH client for your workstation.
3. Start the SSH client.
4. Open a connection to the BIG-IP:

   ```
   ssh -l root [BIG-IP IP address]
   ```

5. Type the root password and press the Enter key.

Addressing general networking issues

You must address several network issues when you place a BIG-IP in your network. These networking issues include routing, DNS configuration, and special e-mail considerations. You need to address these issues based on the type of hardware and software in your network. This section describes the following networking issues:

- **Addressing routing issues**
  There are a variety of routing configuration issues that you need to address. If you did not create a default route with the Setup utility, you must now configure a default route for the BIG-IP. You also must set up routes for the nodes that the BIG-IP manages. You may also want to configure GateD, which allows dynamic routing information to automatically be updated on the BIG-IP.
- **Configuring DNS on the BIG-IP**
  You may need to configure the BIG-IP for DNS resolution or for DNS proxy, and you may even need to convert from rotary or round robin DNS.

- **Configuring email on the BIG-IP**
  There are some special requirements that you need to take into account when configuring email on the BIG-IP.

Addressing routing issues

The BIG-IP must communicate properly with network routers, as well as with the servers, firewalls, and other routers that it manages. Because there is a variety of router configurations, and varying levels of direct control an administrator has over each router, you need to carefully review the router configurations in your own network. You may need to change some routing configurations before you put the BIG-IP into production.

The BIG-IP supports static route configurations, dynamic routing (by way of BGP4, RIP1, RIP2, and OSPF), and subnetting. However, the BIG-IP is also designed to eliminate the need for you to modify routing tables on a router that routes to a BIG-IP. Instead, the BIG-IP uses Address Resolution Protocol (ARP) to notify routers of the IP addresses that it uses on each interface, as well as on its virtual servers.

The following sections address these common routing issues:

- Routing from a BIG-IP to a gateway to the external network
- Routing from content servers to the BIG-IP
- Routing between a BIG-IP to content servers that are on different logical networks
- Setting up dynamic routing with GateD
- Configuring static routes in **/config/routes**

Routing from a BIG-IP to a gateway to the external network

The BIG-IP needs a route to the external network. For most configurations, this should be configured as the default gateway pool on the BIG-IP.

During installation, you were prompted to configure a default route for the BIG-IP. If you need to change the default route at this time, you can set a new default route by editing the default gateway pool.

To change the default route from the Setup utility

1. From the command line, type **config**.
   The Setup utility menu opens.
2. Choose the Default Gateway Pool option.
3. Type the IP address of the gateway you want to add to the default gateway pool.
4. Save and exit.

To change the default route using the Configuration utility

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

2. Click the System tab.
   Look in the **Default Gateway Pool** list for the name of the default gateway pool. Make sure you have the pool name before proceeding to step 3.
3. In the navigation pane, click Pools.
   The Pools screen opens.
4. In the list of pools, click the name of the default gateway pool.
   The pool properties page for that pool opens.
5. In the Resources section of the screen, add or remove gateway IP addresses.
6. Click the **Apply** button.

Routing from content servers to the BIG-IP

The content servers being load balanced by the BIG-IP need to have a default route set to the internal shared floating IP alias of the BIG-IP. For most configurations, this should be configured as the default route on the content server.

For information about setting the default route for your content servers, refer to the product documentation for your server.

Routing between a BIG-IP and content servers on different logical networks

If you need to configure the BIG-IP to use one or more nodes that actually sit on a different logical network from the BIG-IP, you need to assign one or more additional routes to get to those nodes. Set each node's default route so that traffic goes back through the BIG-IP internal interface.

In the following examples, the nodes are on **192.168.6.0/24** and the BIG-IP internal interface is on **192.168.5.0/24**. There are two possible situations which you may have to address:

- **192.168.5.0/24** and **192.168.6.0/24** are on the same LAN (either sharing media or with a switch or hub between them).
- **192.168.5.0/24** and **192.168.6.0/24** are on two different LANs with a router between them.

Case 1: Same LAN

If the nodes are on the same LAN as the BIG-IP, you simply need to add an interface route for **192.168.6.0/24** to an interface on the internal network. You can add this route to the bottom of the **/etc/rc.local** file using the following syntax, where **<ip addr>** is the IP address on the internal interface:

```
route add -net 192.168.6 -interface <ip addr>
```

◆ Note

*You must have the interface defined correctly in the **/etc/hosts** file in order to use this syntax.*

Case 2: Different LANs

If you have nodes on different LANs from the BIG-IP, you need to add a static gateway route on the BIG-IP itself. If, for example, the router that connects the **192.168.5** network and the **192.168.6** network has IP addresses **192.168.5.254** and **192.168.6.254**, then you could use the following command to create the necessary static route on the BIG-IP:

```
route add -net 192.168.6.0 -gateway 192.168.5.254
```

You should add this command to the end of the file **/etc/netstart** so that it runs each time the BIG-IP boots.

You may also need to set the default route on the nodes to point to the router between the LANs. For example:

```
route add default -gateway 192.168.6.254
```

Finally, you need to set the default route on the router between the LANs to the shared alias on the BIG-IP. For example, type the command:

```
route add default -gateway 192.168.5.200
```

◆ Note

These examples assume you are using a UNIX-based router. The exact syntax for your router may be different.

It is not necessary to set the default route for nodes directly to the BIG-IP, as long as the default path eventually routes through the BIG-IP.

Setting up dynamic routing with GateD

The GateD daemon allows the BIG-IP to exchange dynamic routing updates with your routers. Setting up the GateD daemon is a three-part task:

- You need to create the GateD configuration file, **/config/gated.conf.**
- You need to start the GateD daemon.
- You need to edit the **/etc/netstart** file.

◆ Tip

You are not required to configure GateD on the BIG-IP. The BIG-IP can meet most routing requirements without using GateD.

◆ Note

Additional documentation for GateD is available through the web server on the BIG-IP.

To create the GateD configuration file

GateD relies on a configuration file, typically named **/config/gated.conf**, which can be relatively simple, or can be very complex, depending on the routing needs of your network. The BIG-IP web server includes the GateD

online documentation (in the Configuration utility home screen, under the **Online Documentation** section, click **GateD*)***. Note that the GateD configuration guide details the process of creating the GateD configuration file, and also provides samples of common protocol configurations.

To immediately start the GateD daemon on the BIG-IP

Once you create the GateD configuration file, you need to start the GateD daemon on the command line using the following command:

```
bigip# gated
```

Configuring static routes in /config/routes

You can create the file **/config/routes** on the BIG-IP for configuring static route information. The information you add to **/config/routes** is synchronized between units in a BIG-IP redundant pair. When you upgrade, the route information is saved and reinstalled when the upgrade is complete.

You can add routes to **/config/routes** using the format in Figure 12.2.

```
route add -net 10.1.10.0 -netmask 255.255.255.0 -gateway 10.1.30.254
route add -net 10.1.20.0 -netmask 255.255.255.0 -gateway 10.1.30.254
```

Figure 12.2 *Example entries in /config/routes*

Configuring DNS on the BIG-IP

If you plan to use DNS in your network, you can configure DNS on the BIG-IP. There are three different DNS issues that you may need to address when setting up the BIG-IP:

- Configuring DNS resolution on the BIG-IP
- Configuring DNS proxy
- Converting from rotary or round robin DNS

Configuring DNS resolution

When entering virtual addresses, node addresses, or any other addresses on the BIG-IP, you can use the address, host name, or fully qualified domain name (FQDN).

The BIG-IP looks up host names and FQDNs in the **/etc/hosts** file. If it does not find an entry in that file, then it uses DNS to look up the address. In order for this to work, you need to create an **/etc/resolv.conf** file. The file should have the following format:

```
nameserver <DNS_SERVER_1>
search <DOMAIN_NAME_1> <DOMAIN_NAME_2>
```

In place of the **<DNS_SERVER_1>** parameter, use the IP address of a properly configured name server that has access to the Internet. You can specify additional name servers as backups by inserting an additional **nameserver** line for each backup name server.

If you configure the BIG-IP itself as a DNS proxy server, then we suggest that you choose its loopback address (**127.0.0.1**) as the first name server in the **/etc/resolv.conf** file.

Replace the **<DOMAIN_NAME_1>** and **<DOMAIN_NAME_2>** parameters with a list of domain names to use as defaults. The DNS uses this list to resolve hosts when the connection uses only a host name, and not an FQDN. When you enter domain names in this file, separate each domain name with a space, as shown in Figure 12.3.

```
; example /etc/resolv.conf
nameserver 127.0.0.1
nameserver 127.16.112.2  ;ip address of main DNS server
search mysite.com store.mysite.com
```

Figure 12.3 *Sample **/etc/resolv.conf** file*

You can also configure the order in which name resolution checks are made by configuring the **/etc/irs.conf** file. You should set this file so that it checks the **/etc/hosts** file first, and then checks for DNS entries. See Figure 12.4, for an example of how to make the entry in the **/etc/irs.conf** file.

```
hosts          local   continue
hosts          dns
```

Figure 12.4 *Sample entry for the **/etc/irs.conf** file*

Configuring DNS proxy

The BIG-IP is automatically configured as a DNS proxy or forwarder. This is useful for providing DNS resolution for servers and other equipment load balanced by the BIG-IP. This can be set in the Setup utility.

To re-configure DNS proxy, you simply edit the **/etc/named.boot** file that contains these two lines:

```
forwarders <DNS_SERVERS>
options forward-only
```

In place of the **<DNS_SERVERS>** parameter, use the IP addresses of one or more properly configured name servers that have access to the Internet.

You can also configure the BIG-IP to be an authoritative name server for one or more domains. This is useful when DNS is needed in conjunction with internal domain names and network addresses for the servers and other equipment behind the BIG-IP. Refer to the BIND documentation for more details.

Converting from rotary or round robin DNS

If your network is currently configured to use rotary DNS, your node configuration may not need modification. However, you need to modify your DNS zone tables to map to a single IP address instead of to multiple IP addresses.

For example, if you had two Web sites with domain names of **www.SiteOne.com** and **www.SiteTwo.com**, and used rotary DNS to cycle between two servers for each Web site, your zone table might look like the one in Figure 12.5.

```
www.SiteOne.com  IN A 192.168.1.1
                 IN A 192.168.1.2
www.SiteTwo.com  IN A 192.168.1.3
                 IN A 192.168.1.4
```

Figure 12.5 *Sample zone table with two Web sites and four servers*

In the BIG-IP configuration, the IP address of each individual node used in the original zone table becomes hidden from the Internet. We recommend that you use the Internet reserved address range as specified by RFC 1918 for your nodes. In place of multiple addresses, simply use a single virtual server associated with your site's domain name.

Using the above example, the DNS zone table might look like the zone table shown in Figure 12.6.

```
www.SiteOne.com  IN A 192.168.100.231
www.SiteTwo.com  IN A 192.168.100.232
```

Figure 12.6 *Sample zone table with two Web sites and two servers.*

Configuring email

Another optional feature you can set up when you configure the BIG-IP is email. You can configure the BIG-IP to send email notifications to you, or to other administrators. The BIG-IP uses Sendmail as its mail transfer agent. The BIG-IP includes a sample Sendmail configuration file that you can use to start with, but you will have to customize the Sendmail setup for your network environment before you can use it.

Before you begin setting up Sendmail, you may need to look up the name of the mail exchanger for your domain. If you already know the name of the mail exchanger, continue with the following section, *Setting up Sendmail*.

Setting up Sendmail

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

To set up and start Sendmail

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

   ```
   DS<MAILHUB_OR_RELAY>
   ```

3. Save and close the **/config/sendmail.cf** file.
4. If you want to allow Sendmail to flush outgoing messages from the queue for mail that cannot be delivered immediately:
   a) Open the **/config/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
   ```

   b) Save and close the **/config/crontab** file.
5. If you want to prevent returned or undelivered email from going unnoticed:
   a) Open the **/config/aliases** file and create an entry for **root** to point to you or another administrator at your site:

   ```
   root: networkadmin@SiteOne.com
   ```

   b) Save and close the **/config/aliases** file.
   c) Run the **newaliases** command to generate a new aliases database that incorporates the information you added to the **/config/aliases** file.
6. To turn Sendmail on, either reboot the system or type the following command:

   ```
   /usr/sbin/sendmail -bd -q30m
   ```

Using a serial terminal with the BIG-IP

There are a couple of different ways to add a serial terminal to the BIG-IP. You can add a serial terminal in addition to the console, or you can add a serial terminal as the console. The difference between the two is:

- A serial terminal configured as a terminal displays a simple login. You can log in and run commands and edit files. In this case, you can use the serial terminal in addition to the keyboard and monitor.

- A serial terminal configured as the console displays system messages and warnings in addition to providing a login prompt. In this case, the serial terminal replaces the keyboard and monitor.

To connect the serial terminal to the BIG-IP

Connect a serial line cable between the terminal device and the BIG-IP. On the back of BIG-IP is a male, 9-Pin RS232C connector labeled **Terminal**. (Be sure not to confuse this with the fail-over connection which is also a male, 9-pin connector.)

Do not use the fail-over cable to connect the serial terminal to the BIG-IP. A null modem cable is required

The connector is wired as a DTE device, and uses the signals described in Table 12.1.

Pin	Source	Usage
1	External	Carrier detect
2	External	Received data
3	Internal	Transmitted data
4	Internal	Data terminal ready
5	Both	Signal ground
7	Internal	Request to send
8	External	Clear to send

Table 12.1 *Serial line cable signals*

The connector is wired for direct connection to a modem, with receipt of a Carrier Detect signal generating transmission of a login prompt by the BIG-IP. If you are planning to connect to a terminal or to connect a PC and utilize a terminal emulation program such as HyperTerminal™, you need a null modem cable with the wiring to generate the signals shown in Table 12.1.

◆ Note

*You can achieve acceptable operation by wiring pins 7 to 8 and pins 1 to 4 at the back of BIG-IP (and turning hardware flow control **off** in your terminal or terminal emulator).*

Configuring a serial terminal in addition to the console

You can configure a serial terminal for the BIG-IP in addition to the standard console.

To configure the serial terminal in addition to the console

1. Connect the serial terminal to the BIG-IP.
2. Configure the serial terminal settings in your terminal or terminal emulator or modem as follows:
   - 9600 baud
   - 8 bits
   - 1 stop bit
   - No parity
3. Open the **/etc/ttys** file and find the line that reads **tty00 off**. Modify it as shown here:

```
# PC COM ports (tty00 is DOS COM1)
tty00 "/usr/libexec/getty default" vt100 in secure
```

4. Save the **/etc/ttys** file and close it.
5. Reboot the BIG-IP.

Configuring a serial terminal as the console

You can configure the serial terminal as the console.

To configure the serial terminal as the console

1. Disconnect the keyboard from the BIG-IP.
2. Connect the serial terminal to the BIG-IP. When there is no keyboard connected to the BIG-IP, the BIG-IP defaults to using the serial port for the console.
3. Configure the serial terminal settings in your terminal or terminal emulator or modem as follows:
   - 9600 baud
   - 8 bits
   - 1 stop bit
   - No parity
4. Reboot the BIG-IP.

Forcing a serial terminal to be the console

In the case where you have not yet connected the serial terminal or it is not active when the BIG-IP is booted, as it might be if you are using a terminal server or dial-up modem, you can force the controller to use the serial terminal as a console. Note that you do not need to disconnect the keyboard if you use this procedure to force the serial line to be the console.

To force a serial terminal to be the console

1. Edit the **/etc/boot.default** file.
   Find the entry **-console auto**. Change this entry to **-console com**.
2. Save the **/etc/boot.default** file and exit the editor.
3. Plug the serial terminal into the serial port on the BIG-IP.
4. Turn on the serial terminal.
5. Reboot the BIG-IP.

Once you configure a serial terminal as the console for the BIG-IP, the following conditions apply:

Keyboard/monitor access is disabled, and logging in is only possible via Secure Telnet (SSH), if configured, or the serial line.

*If the **boot.default** file is corrupted, the system does not boot at all. Save a backup copy of the original file and keep a bootable CD-ROM on hand.*

*The **boot.default** file must contain either the line: **-console com** or the line: **-console auto**. Do not configure both settings. This could cause problems when you attempt to boot the system.*

Configuring RADIUS or LDAP authentication

You can configure the BIG-IP to use a RADIUS or LDAP server on your network to authenticate users attempting to access the controller with SSH. In this configuration, the RADIUS or LDAP server can function as a central repository of users that are allowed access to the BIG-IP for administrative purposes.

To configure RADIUS login support

Follow these steps to enable RADIUS authentication on BIG-IP.

1. Create the directory **/etc/raddb**.

2. Create the file **/etc/raddb/servers**. Each line should contain the host name of the radius server to connect to, and the secret used by that server (see Figure 12.7). For security reasons, we recommend that you use IP addresses instead of host names for the entries in this file. If you specify a host name for an entry, we recommend that you add the host name to the **/etc/hosts** file.

```
# this is the /etc/raddb/server file
# format is <radius server> <secret>
radius.test.net testing123
```

Figure 12.7 *The location of the secret in* ***/etc/raddb/servers***

3. Edit the **/etc/login.conf** file. Locate these lines at the top of the file. Replace **my_radius_server** with the hostname of your RADIUS server. The hostname you specify must also exist in the **/etc/raddb/servers** file you created in step 2 (see Figure 12.8).

```
radius-defaults:auth=passwd:\
        :auth-ssh=radius,passwd:\
        :radius-server=my_radius_server:
```

Figure 12.8 *The* ***radius-defaults*** *settings for RADIUS authentication*

4. Change the default configuration to include the **radius-default** section like this (see Figure 12.9):

```
default:\
        :path=/bin /usr/bin /usr/contrib/bin:\
        :datasize-cur=16M:\
        :tc=radius-defaults:
```

Figure 12.9 *Example default settings for RADIUS login support*

5. Before logging out, test the configuration by using SSH to connect to the BIG-IP. That way you can correct any configuration errors which could prevent you from logging in to the BIG-IP.

Configuring LDAP login support

To configure the BIG-IP for LDAP authentication, you need to modify the **/etc/login.conf** file. You can configure LDAP authentication on the BIG-IP with LDAP servers that store passwords in encrypted or hashed format, or you can configure the BIG-IP to handle LDAP servers that use plain text passwords.

To configure an LDAP server that stores encrypted passwords

In some LDAP servers, passwords are stored encrypted with DES, or stored as MD5 hashes. On these systems, it is best to bind to the server directly in order to let the LDAP server match the passwords. The **login_ldap** utility can be configured to bind directly to the server with the following settings in the **/etc/login.conf** file.

1. Edit **/etc/login.conf**. Locate these lines at the top of the file. Replace **my_ldap_server** with the host name of your LDAP server. Replace the value for **ldap-basedn** with the appropriate **basedn** for your LDAP server (see Figure 12.10).

```
ldap-defaults:auth=passwd:\
        :auth-ssh=ldap,passwd:\
        :ldap-server=my_ldap_server:\
        :ldap-server-user=cn=Manager,dc=test,dc=net:\
        :ldap-basedn=dc=test,dc=net:\
        :ldap-user-bind=yes:
```

Figure 12.10 *Example* ***ldap-defaults*** *settings for an LDAP server that stores encrypted passwords*

2. Locate the default authentication type. Change the **tc** value to point to the new **ldap-defaults** type (see Figure 12.11).

```
default:\
        :path=/bin /usr/bin /usr/contrib/bin:\
        :datasize-cur=16M:\
        :tc=ldap-defaults:
```

Figure 12.11 *Example default settings for an LDAP server that stores encrypted passwords*

To configure an LDAP server that stores plain text passwords

Other LDAP servers store user passwords in plain text. Because of this, these servers require the root LDAP user to log in to see these users. Use these instructions to configure BIG-IP to authenticate to the server with the root user identity before each user authentication.

1. Edit the **/etc/login.conf** file. Locate the lines at the top of the file after the **auth-bsdi-defaults** type. Replace **my-ldap-server** with the values from your LDAP configuration. Change **ldap-user-bind** to **no**. The **ldap-sever-user** may not be required by your configuration. If it is not, remove that line (see Figure 12.12).

```
ldap-defaults:auth=passwd:\
        :auth-ssh=ldap,passwd:\
        :ldap-server=my_ldap_server:\
        :ldap-server-user=cn=Manager,dc=test,dc=net:\
        :ldap-basedn=dc=test,dc=net:\
        :ldap-user-bind=no:
```

Figure 12.12 *Example excerpt from the* ***/etc/login.conf*** *for an LDAP server that stores plain text passwords*

2. Locate the **default** authentication type. Change the **tc** value to point to the new **ldap-defaults** type (see Figure 12.13).

```
default:\
        :path=/bin /usr/bin /usr/contrib/bin:\
        :datasize-cur=16M:\
        :tc=ldap-defaults:
```

Figure 12.13 *Example change to the* ***tc*** *value*

3. Create the file **/etc/ldapb/servers**. Add one line for the host name of the LDAP server to connect to, and the secret used by that server user (see Figure 12.14). For security reasons, we recommend that you use IP addresses instead of host names for the entries in this file. If you specify a host name for an entry, we recommend that you add the host name to the **/etc/hosts** file.

```
# this is the /etc/ldapdb/server file
# format is <ldap server> <secret>
ldap.test.net secret
```

Figure 12.14 *Where to add the host name of the LDAP server*

4. Before logging out, test the configuration by running the login program, either on a virtual console or using Telnet. That way you can correct any configuration errors before that may prevent you from accessing the BIG-IP.

Allowing multiple authentication styles

We recommend that you allow multiple authentication styles. This allows you to log in even if the LDAP or RADIUS server is not working properly. You can specify multiple authentication styles in the **auth** field such as **radius**,**passwd**. With the example **radius,passwd**, RADIUS is the default authentication style, but you still have the ability to override the style and force a login using the password file by appending **:passwd** after the login

name. This is useful because you need to be able to log in even if the authentication server is down (or if its name gets changed and the **/etc/login.conf** file needs to be updated).

```
# login davidh:passwd
```

or

```
# ssh bigip -l "davidh:passwd"
```

Only the styles that you specify are accepted. For example, **davidh:ldap** would fail, since that style was not specified.

Requiring different authentication styles for different applications

You can configure the BIG-IP authentication system to require different authentication styles for different applications. The following example (see Figure 12.15) would use password authentication by default (at the console), but would require RADIUS for FTP and LDAP for SSH, and would accept RADIUS, LDAP, or network password logins (telnet).

```
my-defaults:\
        :auth-ftp=radius:\
        :radius-server=<my_radius_server>:\
        :auth-ssh=ldap:\
        :ldap-server=<my_ldap_server>:\
        :ldap-basedn=ou=People,dc=<f5>,dc=<com>:\
        :auth-network=radius,ldap,passwd:\
        :auth=passwd:
default:\
        :path=/bin /usr/bin /usr/contrib/bin:\
        :datasize-cur=16M:\
        :tc=my-defaults:
```

Figure 12.15 *Example of password authentication with RADIUS required for other applications*

◆ Note

RADIUS authentication through the BIG-IP is based on the username/password only. It does not support challenge-response authentication methods.

Glossary

Any IP Traffic

Any IP Traffic is a feature that allows the BIG-IP to load balance protocols other than TCP and UDP.

ARL (Akamai Resource Locator)

An ARL is a URL that is modified to point to content on the Akamai Freeflow Network™. In content conversion (akamaization), the URL is converted to an ARL, which retrieves the resource from a geographically nearby server on the Akamai Freeflow Network for faster content delivery.

big3d

The **big3d** utility is a monitoring utility that collects metrics information about paths between a BIG-IP and a specific local DNS server. The **big3d** utility runs on BIG-IP units and it forwards metrics information to 3-DNS Controllers.

BIG-IP active unit

In a redundant system, the active unit is the BIG-IP that currently load balances connections. If the active unit in the redundant system fails, the standby unit assumes control and begins to load balance connections.

BIG-IP web server

The BIG-IP web server runs on a BIG-IP and hosts the Configuration utility.

bigpipe

The **bigpipe** utility provides command line access to the BIG-IP.

BIG/stat

BIG/stat is a statistical monitoring utility that ships on the BIG-IP. This utility provides a snap-shot of statistical information.

BIG/top

BIG/top is a statistical monitoring utility that ships on the BIG-IP. This utility provides real-time statistical information.

BIND (Berkeley Internet Name Domain)

BIND is the most common implementation of DNS, which provides a system for matching domain names to IP addresses.

cacheable content determination

Cacheable content determination is a process that determines the type of content you cache on the basis of any combination of elements in the HTTP header.

cacheable content expression

The cacheable content expression determines, based on evaluating variables in the HTTP header of the request, whether a BIG-IP Cache Controller directs a given request to a cache server or to an origin server. Any content that does not meet the criteria in the cacheable content expression is deemed non-cacheable.

cache pool

The cache pool specifies a pool of cache servers to which requests are directed in a manner that optimizes cache performance. The BIG-IP Cache Controller directs all requests bound for your origin server to this pool, unless you have configured the hot content load balancing feature, and the request is for **hot** (frequently requested) content. See also *hot* and *origin server*.

chain

A chain is a series of filtering criteria used to restrict access to an IP address. The order of the criteria in the chain determines how the filter is applied, from the general criteria first, to the more detailed criteria at the end of the chain.

content affinity

Content affinity ensures that a given subset of content remains associated with a given cache server to the maximum extent possible, even when cache servers become unavailable, or are added or removed. This feature also maximizes efficient use of cache memory.

content converter gateway

A content converter gateway is a gateway for converting URLs to ARLs. See also *ARL*.

content demand status

The content demand status is a measure of the frequency with which content in a given hot content subset is requested over a given hit period. Content demand status is either **hot**, in which case the number of requests for content in the hot content subset during the most recent hit period has exceeded the hot threshold, or **cool**, in which case the number of requests during the most recent hit period is less than the cool threshold. See also c*ool, cool threshold, hit period, hot*, *hot content subset*, and *hot threshold.*

content hash size

Specifies the number of units, or hot content subsets, into which the content is divided when determining whether content is hot or cool. The requests for all content in a given subset are summed, and a state (hot or cool) is assigned to each subset. The content hash size should be within the same order of magnitude as the actual number of requests possible. For example, if the entire site is composed of 500,000 pieces of content, a content hash size of 100,000 is typical.

If you specify a value for hot pool, but do not specify a value for this variable, the cache statement uses a default hash size of 10 subsets. See also *cool, hot,* and *hot content subset.*

content stripes

In products that support caching, content stripes are cacheable content subsets distributed among your cache servers.

cookie persistence

Cookie persistence is a mode of persistence where the BIG-IP stores persistent connection information in a cookie.

cool

Cool describes content demand status when you are using hot content load balancing. See also *content demand status, hot,* and *hot content load balancing.*

cool threshold

The cool threshold specifies the maximum number of requests for given content that will cause that content to change from hot to cool at the end of the hit period.

If you specify a variable for hot pool, but do not specify a value for this variable, the cache statement uses a default cool threshold of 10 requests. See also *cool, hit period,* and *hot*.

default VLANs

The BIG-IP is configured with two default VLANs, one for each interface. One default VLAN is named **internal** and one is named **external**. See also *VLAN*.

default wildcard virtual server

A default wildcard virtual server has an IP address and port number of **0.0.0.0:0**. or ***:*** or **"any":"any"**. This virtual server accepts all traffic that does not match any other virtual server defined in the configuration.

dynamic load balancing

Dynamic load balancing modes use current performance information from each node to determine which node should receive each new connection. The different dynamic load balancing modes incorporate different performance factors such as current server performance and current connection load.

Dynamic Ratio load balancing mode

Dynamic Ratio mode is like Ratio mode (see *Ratio mode*), except that ratio weights are based on continuous monitoring of the servers and are therefore continually changing. Dynamic Ratio load balancing may currently be implemented on RealNetworks RealServer platforms, on Windows

platforms equipped with Windows Management Instrumentation (WMI), or on a server equipped with either the UC Davis SNMP agent or Windows 2000 Server SNMP agent.

dynamic site content

Dynamic site content is site content that is automatically generated each time a user accesses the site. Examples are current stock quotes or weather satellite images.

EAV (Extended Application Verification)

EAV is a health check that verifies an application on a node by running that application remotely. EAV health check is only one of the three types of health checks available on a BIG-IP. See also *health check*, *health monitor* and *external monitor*.

ECV (Extended Content Verification)

ECV is a health check that allows you to determine if a node is up or down based on whether the node returns specific content. ECV health check is only one of the three types of health checks available on a BIG-IP. See also *health check*.

external monitor

An external monitor is a user-supplied health monitor. See also, *health check*, *health monitor*.

external VLAN

The external VLAN is a default VLAN on the BIG-IP. In a basic configuration, this VLAN has the administration ports locked down. In a normal configuration, this is typically a VLAN on which external clients request connections to internal servers.

fail-over

Fail-over is the process whereby a standby unit in a redundant system takes over when a software failure or a hardware failure is detected on the active unit.

fail-over cable

The fail-over cable directly connects the two units together in a redundant system.

Fastest mode

Fastest mode is a dynamic load balancing mode that bases connection distribution on which server currently exhibits the fastest response time to node pings.

FDDI (Fiber Distributed Data Interface)

FDDI is a multi-mode protocol used for transmitting data on optical-fiber cables at speeds up to 100 Mbps.

floating self IP address

A floating self IP address is an additional self IP address for a VLAN that serves as a shared address by both units of a BIG-IP redundant system.

forward proxy caching

Forward proxy caching is a configuration in which a BIG-IP Cache Controller redundant system uses content-aware traffic direction to enhance the efficiency of an array of cache servers storing Internet content for internal users.

health check

A health check is a BIG-IP feature that determines whether a node is **up** or **down**. Health checks are implemented through health monitors. See also *health monitor*, *ECV*, *EAV*, and *external monitor*.

health monitor

A health monitor checks a node to see if it is **up** and functioning for a given service. If the node fails the check, it is marked **down**. Different monitors exist for checking different services. See also *health check*, *EAV*, *ECV*, and *external monitor*.

hit period

The hit period specifies the period, in seconds, over which to count requests for particular content before determining whether to change the state (hot or cool) of the content.

If you specify a value for hot pool, but do not specify a value for this variable, the cache statement uses a default hit period of 10 seconds. See also *cool*, *hot*, and *hot pool*.

host

A host is a network server that manages one or more virtual servers that the 3-DNS Controller uses for load balancing.

hot

Hot is a term used to define frequently requested content based on the number of requests in a given time period for a given hot content subset. See also *hot content subset*.

hot content load balancing

Hot content load balancing identifies hot or frequently requested content on the basis of number of requests in a given time period for a given hot content subset. A hot content subset is different from, and typically smaller than, the

content subsets used for content striping. Requests for hot content are redirected to a cache server in the hot pool, a designated group of cache servers. This feature maximizes the use of cache server processing power without significantly affecting the memory efficiency gained by cacheable content determination. See also *hot*, *hot content subset*, and *hot pool*.

hot content subset

A hot content subset is different from, and typically smaller than, the content subsets used for cacheable content determination. This is created once content has been determined to be hot, and is taken or created from the content subset. See also *cacheable content determination.*

hot pool

A hot pool is a designated group of cache servers to which requests are load balanced when the requested content is hot. If a request is for hot content, the BIG-IP Cache Controller redundant system directs the request to this pool.

hot threshold

The hot threshold specifies the minimum number of requests for content in a given hot content subset that will cause that content to change from cool to hot at the end of the period.

If you specify a value for hot pool, but do not specify a value for this variable, the cache statement uses a default hot threshold of 100 requests. See also *cool*, *hot*, *hot content subset*, and *hot pool*.

HTTP redirect

An HTTP redirect sends an HTTP 302 Object Found message to clients. You can configure a pool with an HTTP redirect to send clients to another node or virtual server if the members of the pool are marked **down**.

ICMP (Internet Control Message Protocol)

ICMP is an Internet communications protocol used to determine information about routes to destination addresses, such as virtual servers managed by BIG-IP units and 3-DNS Controllers.

intelligent cache population

Intelligent cache population allows caches to retrieve content from other caches in addition to the origin web server. Use this feature when working with non-transparent cache servers that can receive requests destined for the cache servers themselves. Intelligent cache population minimizes the load on the origin web server and speeds cache population. See also *non-transparent cache server* and *transparent cache server.*

interface

The physical port on a BIG-IP is called an interface. See also *link*.

internal VLAN

The internal VLAN is a default VLAN on the BIG-IP. In a basic configuration, this VLAN has the administration ports open. In a normal configuration, this is a network interface that handles connections from internal servers.

IPSEC

IPSEC (Internet Security Protocol) is a communications protocol that provides security for the network layer of the Internet without imposing requirements on applications running above it.

iQuery

A UDP based protocol used to exchange information between BIG-IP units and 3-DNS Controllers. The iQuery protocol is officially registered for port 4353.

last hop

A last hop is the final hop a connection took to get to the BIG-IP. You can allow the BIG-IP to determine the last hop automatically to send packets back to the device from which they originated. You can also specify the last hop manually by making it a member of a last hop pool.

Least Connections mode

Least Connections mode is a dynamic load balancing mode that bases connection distribution on which server currently manages the fewest open connections.

link

A link is a physical interface on the BIG-IP connected to another physical interface in a network.

link aggregation

The link aggregation feature allows you to combine a number of links together to act as one interface.

load balancing mode

A particular method of determining how to distribute connections across an array.

loopback adapter

A loopback adapter is a software interface that is not associated with an actual network card. The nPath routing configuration requires you to configure loopback adapters on servers.

MAC (Media Access Control)

MAC is a protocol that defines the way workstations gain access to transmission media, and is most widely used in reference to LANs. For IEEE LANs, the MAC layer is the lower sublayer of the data link layer protocol.

MAC address

A MAC address is used to represent hardware devices on an Ethernet network.

member

Member is a reference to a node when it is included in a particular pool. Pools typically include multiple member nodes.

minimum active members

The minimum active members is the number of members that must be active in a priority group in order for the BIG-IP to send its requests to that group. If the number of active members falls below this number, requests are sent to the next highest priority group (the priority group with the next lowest priority number).

miss request

When a cache does not have requested content and cannot respond to the request, it is called a miss request.

monitor

The BIG-IP uses monitors to determine whether nodes are **up** or **down**. There are several different types of monitors and they use various methods to determine the status of a server or service.

monitor destination IP address or IP address:port

The monitor destination IP address or address:port for a user defined monitor is used mainly for setting up a node alias for the monitor to check. All nodes associated with that monitor will be marked down if the alias node (destination IP address:port) is marked down. See also *node alias*.

monitor instance

You create a monitor instance when a health monitor is associated with a node, node address, or port. It is the monitor instance that actually performs the health check, not the monitor.

monitor template

A monitor template is a system-supplied health monitor that is used primarily as a template to create user-defined monitors, but in some cases can be used as is. The BIG-IP includes a number of monitor templates, each

specific to a service type, for example, HTTP and FTP. The template has a template type that corresponds to the service type and is usually the name of the template.

named

Named is the name server utility, which manages domain name server software.

NAT (Network Address Translation)

A NAT is an alias IP address that identifies a specific node managed by the BIG-IP to the external network.

node

A node is a specific combination of an IP address and port (service) number associated with a server in the array that is managed by the BIG-IP.

node address

A node address is the IP address associated with one or more nodes. This IP address can be the real IP address of a network server, or it can be an alias IP address on a network server.

node alias

A node alias is a node address that the BIG-IP uses to verify the status of multiple nodes. When the BIG-IP uses a node alias to check node status, it pings the node alias. If the BIG-IP receives a response to the ping, it marks all nodes associated with the node alias as **up**. If the BIG-IP does not receive a response to the ping, the it marks all nodes associated with the node alias as **down**.

node port

A node port is the port number or service name that is hosted by a specific node.

node status

Node status indicates whether a node is **up** and available to receive connections, or **down** and unavailable. The BIG-IP uses the node ping and health check features to determine node status.

non-cacheable content

Non-cacheable content is content that is not identified in the cacheable content condition part of a cache rule statement.

non-transparent cache server

Cache servers that can receive requests that are destined for the cache servers themselves are called non-transparent cache servers.

Observed mode

Observed mode is a dynamic load balancing mode that bases connection distribution on a combination of two factors: the server that currently hosts the fewest connections and also has the fastest response time.

origin pool

The origin pool specifies a pool of servers that contain original copies of all content. Requests are load balanced to this pool when any of the following is true: the requested content is not cacheable, no cache server is available, or the BIG-IP Cache Controller redundant system is redirecting a request from a cache server that did not have the requested content.

origin server

An origin server is the web server on which all original copies of your content reside.

performance monitor

A performance monitor gathers statistics and checks the state of a target device.

persistence

A series of related connections received from the same client, having the same session ID. When persistence is turned **on**, a BIG-IP sends all connections having the same session ID to the same node, instead of load balancing the connections.

pool

A pool is composed of a group of network devices (called members). The BIG-IP load balances requests to the nodes within a pool based on the load balancing method and persistence method you choose when you create the pool or edit its properties.

port

A port can be represented by a number that is associated with a specific service supported by a host. Refer to the Services and Port Index for a list of port numbers and corresponding services.

port-specific wildcard virtual server

A port-specific wildcard virtual server is a wildcard virtual server that uses a port number other than **0**. See *wildcard virtual server*.

port mirroring

Port mirroring is a feature that allows you to copy traffic from any port or set of ports to a single, separate port where a sniffing device is attached.

Predictive mode

Predictive mode is a dynamic load balancing mode that bases connection distribution on a combination of two factors: the server that currently hosts the fewest connections, and also has the fastest response time. Predictive mode also ranks server performance over time, and passes connections to servers which exhibit an improvement in performance rather than a decline.

rate class

You create a rate filter from the Configuration utility or command line utility. When you assign a rate class to a rate filter, a rate class determines the volume of traffic allowed through a rate filter. See also *rate filter*.

rate filter

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

ratio

A ratio is a parameter that assigns a weight to a virtual server for load balancing purposes.

Ratio mode

The Ratio load balancing mode distributes connections across an array of virtual servers in proportion to the ratio weights assigned to each individual virtual server.

receive expression

A receive expression is the text string that the BIG-IP looks for in the web page returned by a web server during an extended content verification (ECV) health check.

redundant system

Redundant system refers to a pair of BIG-IP units that are configured for fail-over. In a redundant system, there are two units, one running as the active unit and one running as the standby unit. If the active unit fails, the standby unit takes over and manages connection requests.

remote administrative IP address

A remote administrative IP address is an IP address from which a BIG-IP allows shell connections, such as Telnet or SSH.

remote server acceleration

A remote server acceleration configuration is a configuration in which a BIG-IP Cache Controller redundant system uses content-aware traffic direction to enhance the efficiency of an array of cache servers that cache content for a remote web server.

RFC 1918 addresses

An RFC 1918 address is an address that is within the range of non-routable addresses described in the IETF RFC 1918.

Round Robin mode

Round Robin mode is a static load balancing mode that bases connection distribution on a set server order. Round Robin mode sends a connection request to the next available server in the order.

self IP address

Self IP addresses are the IP addresses owned by the BIG-IP that you use to access the internal and external VLANs.

send string

A send string is the request that the BIG-IP sends to the web server during an extended content verification (ECV) health check.

service

Service refers to services such as TCP, UDP, HTTP, and FTP.

Setup utility

The Setup utility walks you through the initial system configuration process. You can run the Setup utility from either the command line or the Configuration utility start page.

SNAT (Secure Network Address Translation)

A SNAT is a feature you can configure on the BIG-IP. A SNAT defines a routable alias IP address that one or more nodes can use as a source IP address when making connections to hosts on the external network.

SNAT automap

This feature allows the BIG-IP to perform a SNAT automatically on any connection that is coming from the unit’s internal VLAN. It is easier to use than traditional SNATs, and solves certain problems associated with the traditional SNAT.

SNMP (Simple Network Management Protocol)

SNMP is the Internet standard protocol, defined in STD 15, RFC 1157, developed to manage nodes on an IP network.

source processing

Source processing means that the interface rewrites the source of an incoming packet.

spanning tree protocol (STP)

Spanning tree protocol is a protocol that provides loop resolution in configurations where one or more external switches is connected in parallel with the BIG-IP.

SSL gateway

An SSL gateway is a gateway for decrypting HTTP requests to an HTTP server and encrypting the reply.

standby unit

A standby unit in a redundant system is a unit that is always prepared to become the active unit if the active unit fails.

stateful site content

Content that maintains dynamic information for clients on an individual basis and is commonly found on e-commerce sites is called stateful site content. For example, a site that allows a user to fill a shopping cart, leave the site, and then return and purchase the items in the shopping cart at a later time has stateful site content which retains the information for that client's particular shopping cart.

state mirroring

State mirroring is a feature on the BIG-IP that preserves connection and persistence information in a BIG-IP redundant system.

static load balancing modes

Static load balancing modes base connection distribution on a pre-defined list of criteria; it does not take current server performance or current connection load into account.

static site content

Static site content is a type of site content that is stored in HTML pages, and changes only when an administrator edits the HTML document itself.

sticky mask

A sticky mask is a special IP mask that you can configure on the BIG-IP. This mask optimizes sticky persistence entries by grouping more of them together.

tagged VLAN

You can define any interface as a member of a tagged VLAN. You can create a list of VLAN tags or names for each tagged interface.

transparent cache server

A transparent cache server can intercept requests destined for a web server, but cannot receive requests.

transparent node

A transparent node appears as a router to other network devices, including the BIG-IP.

trunk

A trunk is a combination of two or more interfaces and cables configured as one link. See also *link aggregation.*

user-defined monitor

A user-defined monitor is a custom monitor configured by a user, based on a system-supplied monitor template. For some monitor types, you must create a user-defined monitor in order to use them. For all monitor types, you must create a user-defined monitor to change system supplied monitor default values.

virtual address

A virtual address is an IP address associated with one or more virtual servers managed by the BIG-IP.

virtual port

A virtual port is the port number or service name associated with one or more virtual servers managed by the BIG-IP. A virtual port number should be the same TCP or UDP port number to which client programs expect to connect.

virtual server

Virtual servers are a specific combination of virtual address and virtual port, associated with a content site that is managed by a BIG-IP or other type of host server.

VLAN

VLAN stands for virtual local area network. A VLAN is a logical grouping of network devices. You can use a VLAN to logically group devices that are on different network segments.

VLAN name

A VLAN name is the symbolic name used to identify a VLAN. For example, you might configure a VLAN named marketing, or a VLAN named development. See also *VLAN.*

watchdog timer card

A watchdog timer card is a hardware device that monitors the BIG-IP for hardware failure.

wildcard virtual server

A wildcard virtual server is a virtual server that uses an IP address of **0.0.0.0**, ***** or **"any"**. A wildcard virtual server accepts connection requests for destinations outside of the local network. Wildcard virtual servers are included only in Transparent Node Mode configurations.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

T

U

V

W

X