BIG-IP® Reference Guide

version 4.5

MAN-0044-01

Legal Notices

Copyright

Trademarks

Patents

This product protected by U.S. Patent 6,374,300; Pending U.S. Patent 20020040400. Other patents pending.

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

Part I
Introduction to the BIG-IP System

Introduction

1
BIG-IP System Overview

Part II
The Base Network

2
Using the Setup Utility

3
Post-Setup Tasks

Part III
The High-Level Network

4
Pools

5
iRules

18
Configuring SNMP

A
bigpipe Command Reference

B
bigdb Configuration Keys

C
Configuration Files

Glossary

Index

PART I
INTRODUCTION TO THE BIG-IP SYSTEM

Introduction

- IMPORTANT HARDWARE INFORMATION
- Getting started
- Using the Administrator Kit
- What's new in version 4.5
- 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 system, 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 to the ***Configuration Worksheet*** and ***Platform Guide*** for hardware installation instructions, and then refer to the ***BIG-IP Solutions Guide*** to follow the steps for setting up your chosen solution.

Choosing a configuration tool

The BIG-IP system 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, *Using the Setup Utility*.

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 system. Once you complete the instructions for the Setup utility described in this guide, you can use the Configuration utility to perform additional 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 MIBs or the SSH client. The Configuration utility requires Netscape Navigator version 4.7, or Microsoft Internet Explorer version 5.0 and 5.5.

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 system, 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 system console, or you can run commands using a remote shell, such as the SSH client or a Telnet client. For detailed information about the **bigpipe** command line syntax, see Appendix A, *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 system. The information is organized into the guides described below. The following printed documentation is included with the BIG-IP unit.

- **Configuration Worksheet**
  This worksheet provides you with a place to plan the basic configuration for the BIG-IP system.

The following guides are available in PDF format from the CD-ROM provided with the BIG-IP system. 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 system.

- **Platform Guide**
  This guide includes information about the BIG-IP unit. It also contains important environmental warnings.
- **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 system. 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 system 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.
- **BIG-IP Link Controller Solutions Guide**
  This guide provides examples of common link load balancing solutions using the Link Controller. Before you begin installing the hardware, we recommend that you browse this guide to find the load balancing solution that works best for you.

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 system 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 Appendix *A, bigpipe Command Reference.*

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 Introduction.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 Introduction.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 system displays the syntax and usage associated with the command.
- **Third-party documentation for software add-ons**
  The BIG-IP distribution CD contains online documentation for all third-party software.
- **Technical support through the World Wide Web**
  The Dell Support website at **support.dell.com** provides the latest technical documentation.

◆ Note

All references to hardware platforms in this guide refer specifically to systems supplied by F5 Networks, Inc. If your hardware was supplied by another vendor and you have hardware-related questions, please refer to the documentation from that vendor.

What's new in version 4.5

The BIG-IP system offers the following major new features in version 4.5, in addition to many smaller enhancements.

Enhanced support for managing SSL connections

This release includes several new features designed to further simplify the administration of SSL connections. These features include extensive web-based screens for centralized key management, and support for certificate revocation lists (CRLs).

Another new SSL feature is the ability for an SSL proxy to interoperate with an LDAP database to authorize users based on client certificates. This LDAP database can reside either locally on the BIG-IP system, or remotely on another server on your network.

Lastly, you can now limit the number of connections coming into an SSL proxy, for security or load balancing reasons.

For more information on managing SSL connections, see Chapter 7 of the *BIG-IP Reference Guide*.

Easy system account creation

With this release, the BIG-IP system now offers a centralized Setup screen to set the passwords for the three system accounts: **root**, **admin**, and **support**. For the **support** account, you can also specify whether to allow command line access, Web access, or both.

For more information on managing user accounts, see Chapter 15, *Administering the BIG-IP System*.

Security enhancements

You can now use the Setup utility to configure a remote LDAP or RADIUS authentication server. With this feature, you no longer need to directly edit configuration files to set up your LDAP or RADIUS authentication server. For more information about configuring remote authentication, see Chapter 2, *Using the Setup Utility*.

Also, this release of the BIG-IP system expands the number of user roles that you can assign to user accounts for the purpose of user authorization. In addition to the standard Full Read/Write, Partial Read/Write, and Read-Only access levels, you can now choose from three additional access levels. These access levels define which of the three interfaces an administrator can use to access the BIG-IP system (the Configuration utility, the command line interface, or the iControl interface). These user authorization roles are stored in the local LDAP database on the BIG-IP system and are designed to operate in concert with centralized LDAP and RADIUS authentication. For more information on managing user accounts, see Chapter 15, *Administering the BIG-IP System*.

Other useful security features in this release are intrusion detection and protection from denial-of-service attacks. This release includes two new features to assist in detecting network intruders--VLAN mirroring and clone pools. By enabling a clone pool, any traffic directed to a pool is automatically sent to a node within a replicated pool. The release also includes two new global variables to define high water and low water marks, for the adaptive reaping of connections. For more information VLAN mirroring and clone pools, see Chapter 3, *Post-Setup Tasks*, *VLANs*, and Chapter 4, *Pools*.

Universal Inspection Engine

The Universal Inspection Engine (UIE) allows you to apply business decisions to applications and web servics, and provides granular control for switching, persistence, and application level security. The BIG-IP system version 4.5 has the ability to read all HTTP or TCP content.

- **Universal content switching**
  Through a number of new rule elements, such as a set of functions and the variables **http_content** and **tcp_content**, you can now write expressions within rules that search not only HTTP headers, but also HTTP and TCP data content to make load balancing decisions. As part of the new ***iRules*** syntax, these new variables and functions significantly enhance your ability to select the pools that most suit your traffic management needs.
- **Universal persistence**
  Universal persistence allows you to persist on any string within a packet, or persist directly on a specific pool member. You can enable universal persistence by including rules-syntax expressions within a pool definition. In this way, a pool can perfom load-balancing operations such as sending traffic to a specific node within the pool, or load-balancing traffic based on any string or node that you define. Furthermore, the rules syntax has been expanded to allow rules to intelligently persist requests to cache servers based on more granular information in a request.

Universal persistence is particularly useful for persisting HTTP or TCP content that is unique to your application. Examples of universal persistence are for i-mode phone users and for working with BEA Weblogic servers by creating persistence maps on BEA Weblogic identifiers. For more information about the Universal Inspection Engine and iRules, see Chapter 5, *iRules*.

Other rule enhancements

In addition to the new rule functions and variables designed for universal content switching, the rules syntax has been further expanded to include two new rule statements, **log** and **accumulate**.

Furthermore, you can now store your class lists externally instead of within the **bigip.conf** file. Storing your class lists externally improves performance and allows for incremental updates to those lists. To support this feature, you can store external class lists using either the Configuration utility or the iControl interface. For more information about these new functions, see Chapter 5, *iRules*.

Enhanced support for global variables

A number of new global variables are included in this release, such as variables that define high water and low water marks for the adaptive reaping of connections to prevent denial-of-service attacks. Also, the Configuration utility now shows all global variables and presents them in categories, according to function. For more information about these global variables, see Appendix A, *bigpipe Command Syntax*.

RealServer plug-in for UNIX systems

With this release comes support for RealSystem® Server systems running on the UNIX operating system. This feature provides the ability to dynamically load balance and monitor UNIX systems that are running the RealSystem Server application. Once you have compiled and installed the plug-in, you can set up your pool for dynamic load balancing, and create a health monitor to monitor the traffic load on the RealSystem Server system. For more information about the RealSystem Server plug-in, see the ***BIG-IP Reference Guide,*** Chapter 11, *Monitors*.

New health monitor features

This release includes a new EAV health monitor, **udp**, which allows you to check the status of UDP connections. Also, the **reverse** attribute, which marks a node as **down** based on a received string, is now available for the **https** and **https_443** monitors. For more information about these monitors, see Chapter 11, *Monitors*.

Other load balancing enhancements

This release includes several new load balancing features, including enhanced administration of load-balanced connections. For example, through the Configuration utility, **bigpipe** command, or **bigapi**, you can now dump connections verbosely, or configure a timeout for idle HTTP connections. Also, by writing rule-type expressions within pool definitions, you can cause a pool to send a connection directly to one of its pool members. For more information these features, see Chapter 5, *iRules* and Chapter 4, *Pools*.

Support for Link Controller

This release of the BIG-IP system includes an add-on Link Controller module for all BIG-IP HA systems. This module includes such features as support for single routers with multiple IP addresses and uplinks, full duplex billing support, and support for multiple outbound router pools. Also included is a significantly enhanced Web user interface, designed to ease basic link-controller configuration steps and provide more detailed statistics information.

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 systems, 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 system to the BIG-IP HA software, which supports all BIG-IP features.

- **The BIG-IP system**
  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 Link Controller**
  The complete version of the BIG-IP 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 system 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 System Overview

- Introducing the BIG-IP system
- What is a BIG-IP system?
- Configuration
- Monitoring and administration
- The BIG-IP system user interface

Introducing the BIG-IP system

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

This chapter is organized as follows:

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

What is a BIG-IP system?

The BIG-IP system 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 system 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, see the ***BIG-IP Solutions Guide***, Chapter 1, *Overview*.)

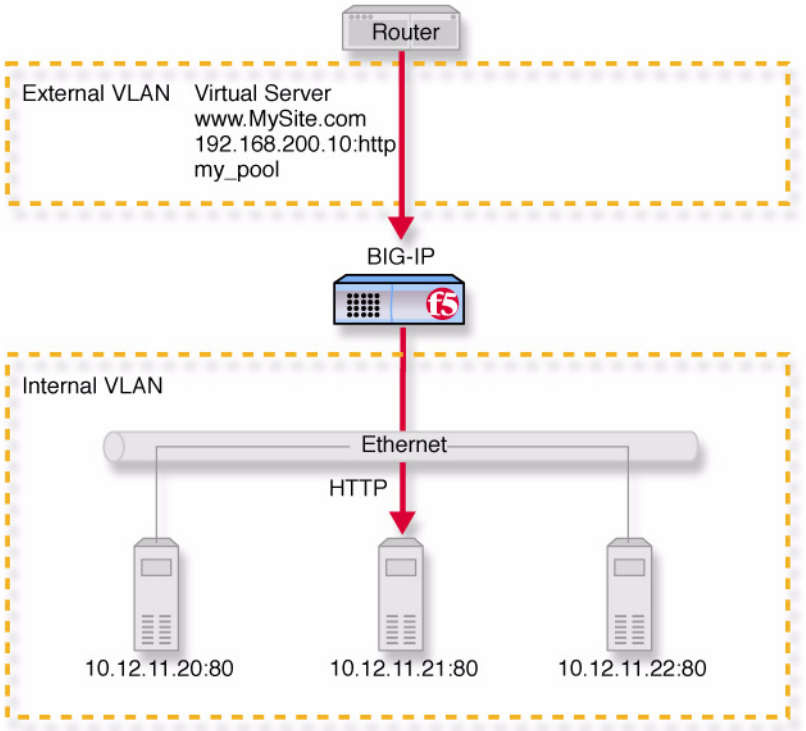

Figure 1.1 *A basic configuration*

Insertion of the BIG-IP system, 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 system 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 system. 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 system address as the default route. By default, the virtual server translates the destination address of the incoming packet to that of the destination network device, making it the source address of the reply packet. The reply packet returns to the BIG-IP system as the default route, and the BIG-IP system 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.)

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

Configuration

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

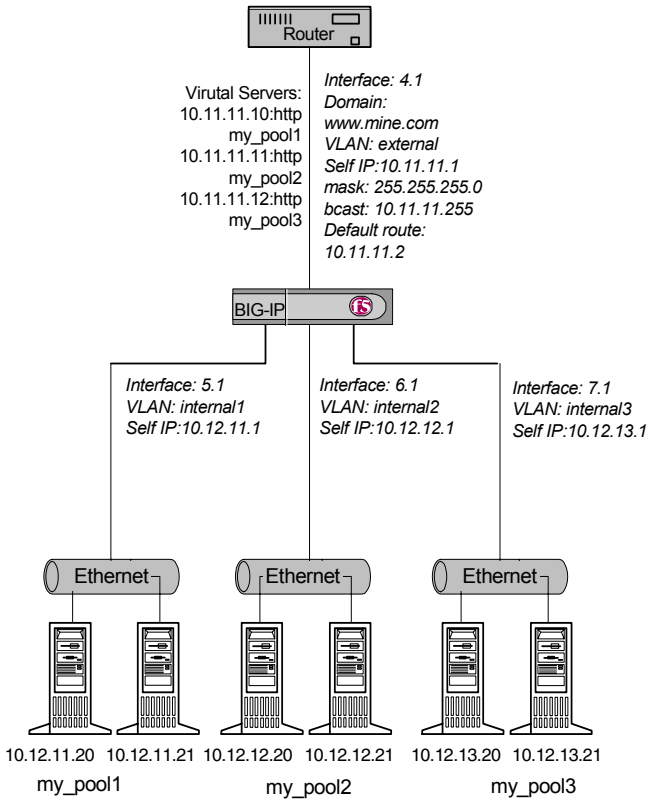

Figure 1.2 *Hardware configuration with base and high-level networks superimposed.*

Hardware configuration

The hardware configuration includes all physical devices and connections in Figure 1.2. That is, the configuration includes the entire physical network. In this case, the configuration consists of a BIG-IP system 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 system 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 a minimum, the two default VLANs (external and internal), domain names, and self IP addresses with netmask and broadcast addresses. Among other things, this base configuration enables you to access the BIG-IP system from a remote host using SSH or HTTPS and in this way gain access to both the command line interface and the browser-based Configuration utility.

At this point, you might want to further configure the base network by performing tasks such as changing settings, 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 use 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 system 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 system 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.

Global settings and filters

Global settings and filters are part of the configuration that belongs 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 the description of the **bigpipe global** command, in Appendix A, *bigpipe Command Reference*.

Filters include IP and Rate filters, and are covered in Chapter 12, *Filters*.

Monitoring and administration

Monitoring and administration refer to the day-by-day tasks of observing traffic, gathering statistics, managing BIG-IP user accounts, 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.

The BIG-IP system user interface

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

The Configuration utility

The Configuration utility resides in the BIG-IP system internal web server. You can access it through the administrative interface on the BIG-IP system using Netscape Navigator version 4.7, or Microsoft Internet Explorer version 5.0 or 5.5.

The Configuration utility shown in Figure 1.3 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 **System** on the navigation pane.)

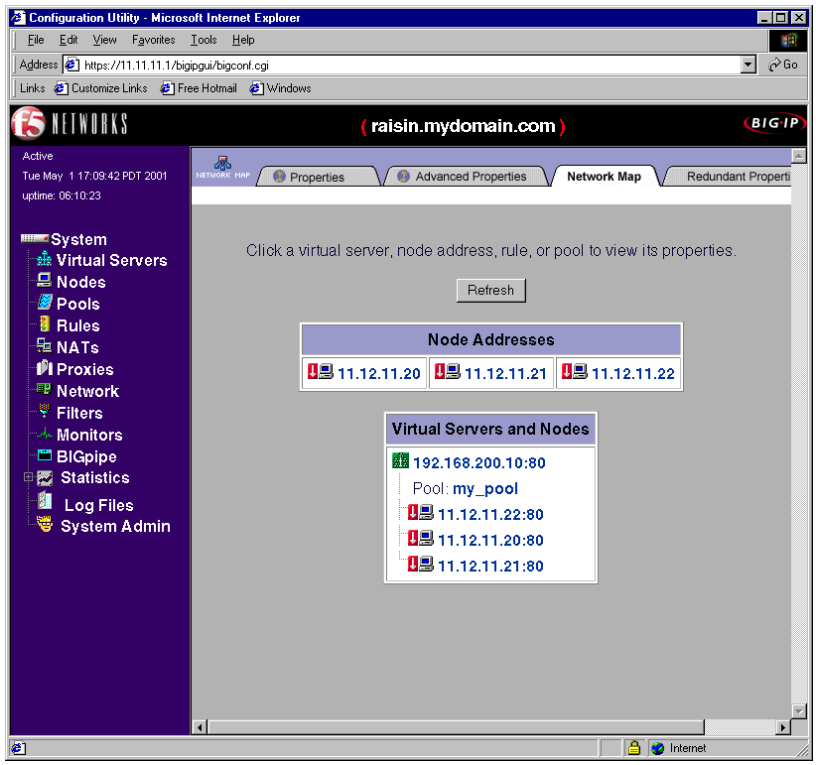

Figure 1.3 *Configuration utility System screen*

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**).

The bigpipe command line interface

You can access the **bigpipe** command line utility on a BIG-IP system with connections for a monitor and keyboard. For a system without a monitor and keyboard attached, 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 }
```

(Note that 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 which utility you use to configure a pool, virtual server, proxy, or other BIG-IP object, the configuration data is entered into the configuration file **/config/bigip.conf**. This produces an entry in that file like the one shown in Figure 1.4. 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 1.4 *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**.

PART II
THE BASE NETWORK

2

Using the Setup Utility

- Creating the initial software configuration with the Setup utility
- Connecting to the BIG-IP system 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 and obtain a license, 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 system 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 system. 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 skips 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, *Post-Setup Tasks*.

The license file installed on the system must be compatible with the latest version of the BIG-IP software before you run the Setup utility. If it is not, you must update the license using the registration key provided to you by your vendor. If you do not have a registration key, please contact your vendor to obtain one. If you choose to continue without obtaining a license, the BIG-IP software will not be fully functional.

Connecting to the BIG-IP system 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. If you run the utility from the console, no reboot is necessary; if you run the utility from the web, the unit reboots automatically; if you run the utility from an SSH client, we recommend that you reboot the unit after you complete the setup. This reboot automatically removes the default IP address and root password provided specifically for the purposes of running the Setup utility remotely. The BIG-IP software 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 **setup**.

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 software 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 software 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 system, 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 system, 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 system 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 address in use on the unit. For information about adding a static IP address to a Microsoft Windows operating system, please refer to the 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 system and you turn the system on, the BIG-IP software 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 software 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 software 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 **Setup Utility**.
5. Fill out each screen using the information from the Setup utility configuration list. After you complete the Setup utility, the BIG-IP system 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.

   ```
   setup
   ```

3. Fill out each screen using the information from the Configuration worksheet. After you complete the Setup utility, the BIG-IP system 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 Chapter 3, *Post-Setup Tasks*, for information on downloading the SSH client from the BIG-IP system.)
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.

   ```
   setup
   ```

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

   ```
   reboot
   ```

Note

*You can rerun the Setup utility at any time using the **setup** 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 to use with the BIG-IP system. 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 are enabled.

◆ Note

You may change your product selection at a later time by running the 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 system 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 FQDN must be less than or equal to 256 characters, but not less than 1 character. Each label part of the name must be 63 characters or fewer. Only letters, numbers, and the characters underscore (_), dash (**-**), and period (**.**) are allowed. For example:

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

*You should only change the host name of the system with the Setup utility. Editing **/etc/hosts**, or using the **hostname** command to change the host name renders the system inaccessible.*

Redundant system settings

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

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 that takes over if the current unit fails. Type in the IP address configured on the internal interface of the other BIG-IP unit in the redundant system.

Fail-over type

There are two types of fail-over to choose from: hard-wired fail-over, and network fail-over. Choose hard-wired fail-over if you plan to connect the units together with the fail-over cable provided with the redundant system. Choose network fail-over if you plan to use the network that the units are connected to for fail-over functionality.

◆ Note

Hard-wired fail-over is only available if the platform supports hard-wired fail-over.

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 system 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 fewer interface devices than you expected, a 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 VLANs to create the BIG-IP system configuration.

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

- Port Lockdown settings
- IP address, netmask, and broadcast
- Floating self IP address, netmask, and broadcast

Note

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 that you want to use on the BIG-IP system, 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 system 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 system load balances.

Associating the primary IP address and VLAN with the host name

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

Configuring a default gateway pool

If a BIG-IP system 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. The first address you type becomes the default route. 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 that 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 system.

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 password for the **admin** user. If SSL is available, the configuration also generates authentication certificates.

◆ Note

If the host name portion of the FQDN is greater than 64 characters, the BIG-IP software cannot use it for the web server FQDN.

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 password for the **admin** user account.
- After you type a password for the **admin** user account, you have the option to type the IP addresses from which web-interface connections are allowed.
- After you type the IP addresses that are allowed to access the unit with the **admin** account, 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 should only add users, or change passwords for existing users, through the Configuration utility.

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****.*

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, RSH, and FTP instead.

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 system will accept administrative connections (either remote shell connections, or connections to the web server on the BIG-IP system). 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 unit.

Setting support access

Next, the Setup utility prompts you to set up a support access account. If you would like to activate a support access account to allow your vendor access to the BIG-IP unit, type a password for the support account. Next, select the access type you want for the support account.

Setting the time zone

Next, you need to specify your time zone. This ensures that the clock for the BIG-IP system 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 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 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 the network load-balanced by the BIG-IP system 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 system. For more information on DNS proxy forwarding, see Chapter 3, *Post-Setup Tasks*.

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

Configuring user authentication

When you run the Setup utility, you can configure authentication for BIG-IP user accounts either through an external LDAP or RADIUS server, or locally on the BIG-IP system. These two authentication options are described following.

Note

*The **root** and **admin** accounts are always authenticated locally.*

Using the local LDAP database only

When you run the Setup utility, you are not required to configure an external LDAP or RADIUS database to manage user authentication. Instead, you can use the default authentication mechanism, which is the BIG-IP system's local LDAP database. In this case, the local LDAP database manages not only authorization for your BIG-IP users, but also authentication. All users subsequently attempting to log on to a BIG-IP system must enter a user name and password, which are checked against user data stored in the local database. If the user name and password are found and verified in that database, the user is authenticated.

Configuring the unit to use an external LDAP or RADIUS server

When you run the Setup utility, you can configure an external (remote) server, either LDAP or RADIUS, to manage user authentication for the BIG-IP system. When you choose this configuration option, all users subsequently attempting to log on to a BIG-IP system must enter a user name and password, which are checked against user data stored in that external database. If the user name and password are found and verified in that database, the user is authenticated.

Note

*In the event that authentication fails with an external LDAP or RADIUS server, you can login with accounts locally such as the **root** and **admin** accounts.*

Configuring external LDAP authentication

When you configure the unit to use an external LDAP server for user authentication you need the following information:

- The IP address of the LDAP server, or the IP address of the primary server if you have more than one LDAP server.
- The base distinguished name of each LDAP server. This name must be the same for each server.
- Optionally, the user name of the account that you want to bind to the LDAP server as the search account. The search account is a read-only account used to do searches. This account must be able to access passwords. If you have more than one LDAP server, this account must be the same on each server.
- If you configure an LDAP search account, you need the password for the that account. If you have more than one LDAP server, you must use the same search account and password.
- After you configure external authentication, you need to set the authorization level, or role, for each user you want to allow to access the controller. You can do this after you complete the Setup utility. Add an account and role for each user in the User Administration screen of the Configuration utility. Since the external authentication server handles the password authentication, you do not need to enter a password for these users. For detailed instructions on setting roles for users, see *Managing user accounts*, on page 17-25.

Configuring external RADIUS authentication

When you configure the unit to use an external RADIUS server for user authentication you need the following information:

- The IP address of the RADIUS server, or the IP address of the primary server and secondary server if you have more than one RADIUS server.
- The port configured for RADIUS traffic on your RADIUS server. Typically, the port configured for RADIUS is port **1645**, the traditional RADIUS port, or port **1812**, the new official RADIUS port.
- The primary RADIUS secret, and if you have a secondary RADIUS server, the secondary RADIUS secret.
- After you configure external authentication, you need to set the authorization level, or role, for each user you want to allow to access the controller. You can do this after you complete the Setup utility. Add an account and role for each user in the User Administration screen of the Configuration utility. Since the external authentication server handles the password authentication, you do not need to enter a password for these users. For detailed instructions on setting roles for users, see *Managing user accounts*, on page 17-25.

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:

```
setup
```

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 system (some versions of the BIG-IP software)

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
- Remote authentication
- License activation
- Configure remote access (for configuration synchronization)
- Set support access

```
lqq I N I T I A L   S E T U P   M E N U qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
x                                                                              x
x    Choose the desired configuration function from the list below.            x
x                                                                              x
x    (A) Configure all services         (R) Steps for redundant systems        x
x                                                                              x
x    REQUIRED                                                                  x
x    (E) Set default gateways           (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    (C) Remote authentication          (O) Configure remote access            x
x    (D) Configure DNS                  (S) Configure SSH                      x
x    (F) Configure FTP                  (T) Configure Telnetd                  x
x    (I) Initialize iControl portal     (U) Configure RSH                      x
x    (K) Set keyboard type              (Y) Set support access                 x
x    (L) License Activation             (Z) Set time zone                      x
x    (M) Define time servers            (Q) Quit                               x
x                                                                              x
x                                                                              x
x    Enter Choice:                                                             x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
```

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 Telnet
- Configuring FTP

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 unit 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 system does not have to do a DNS lookup.

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

- The security mode of the portal. You can allow the portal to handle non-secure requests.
- The name of the Portal object reference file.
- 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 system. 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 Telnet

Use this option to configure the Telnet server only on a BIG-IP system. 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 system. 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 system.

Configuring FTP

Use this option to configure FTP on the BIG-IP system. The Setup utility prompts you for an IP address from which administrators may access the BIG-IP system 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 system.

3

Post-Setup Tasks

- Introducing post-setup tasks
- Interfaces
- VLANs
- Self IP addresses
- Defining additional host names
- Managing the SSH Console
- Addressing general networking issues
- Using a serial terminal with the BIG-IP system
- Trunks
- Spanning Tree Protocol (STP)
- Port Mirroring

Introducing post-setup tasks

Setting up the base network for the BIG-IP system means configuring elements such as the BIG-IP system 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.
- **Additional host names**
  You can insert additional host names and IP addresses for network devices into the **/etc/hosts** file. For example, you can insert host names for the IP addresses that you will assign to virtual servers, and host names for standard devices such as your routers, network interface cards, and servers.
- **SSH console**
  Configuring an SSH console gives you the ability to use a command line interface to securely manage your local BIG-IP system.
- **General networking**
  You can configure a default route, as well as dynamic routing, DNS, and email.
- **Serial terminals**
  You can add a serial terminal in addition to the console, or you can add a serial terminal as the console.

If the BIG-IP system is an IP Application Switch, you also have three other BIG-IP system 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 system Application Switch to a single, separate interface. Typically you would install a sniffer device on the target port for debugging and/or monitoring.

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).

Interfaces

A BIG-IP system 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 system 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.

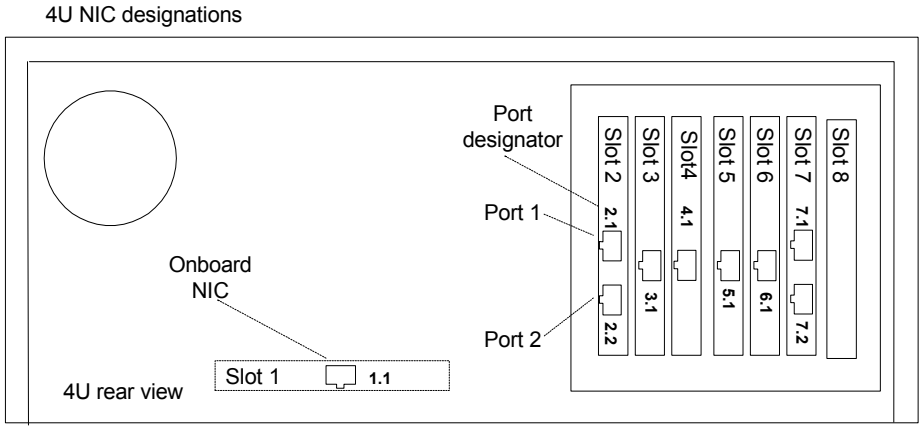

Figure 3.1 *Vertical slot and port numbering*

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

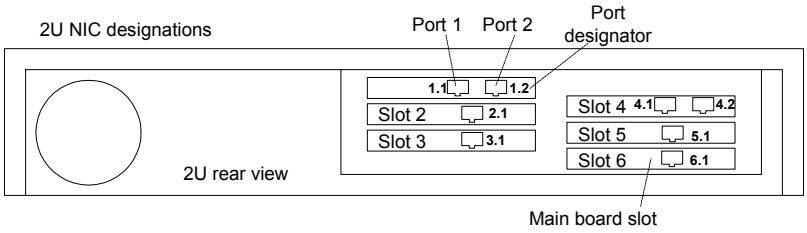

Figure 3.2 *Horizontal slot and port numbering*

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
```

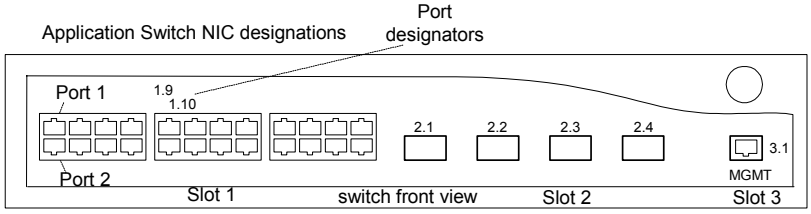

Figure 3.3 *Application Switch slot and port numbering*

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.

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 system is inter-operating with an external switch, the media setting should match that of the switch.

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 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
```

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.

The BIG-IP system 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	You can use the Setup utility to create a default VLAN configuration.
Create, rename, or delete VLANs	You can 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	You can 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.
Configure VLAN mirroring	You can configure the BIG-IP system to replicate packets received by a VLAN and send them to another VLAN or set of VLANs.

Table 3.2 *Configuration options for VLANs*

Default VLAN configuration

By default, the Setup utility configures each interface on the BIG-IP system as a member of a VLAN. The BIG-IP system 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.

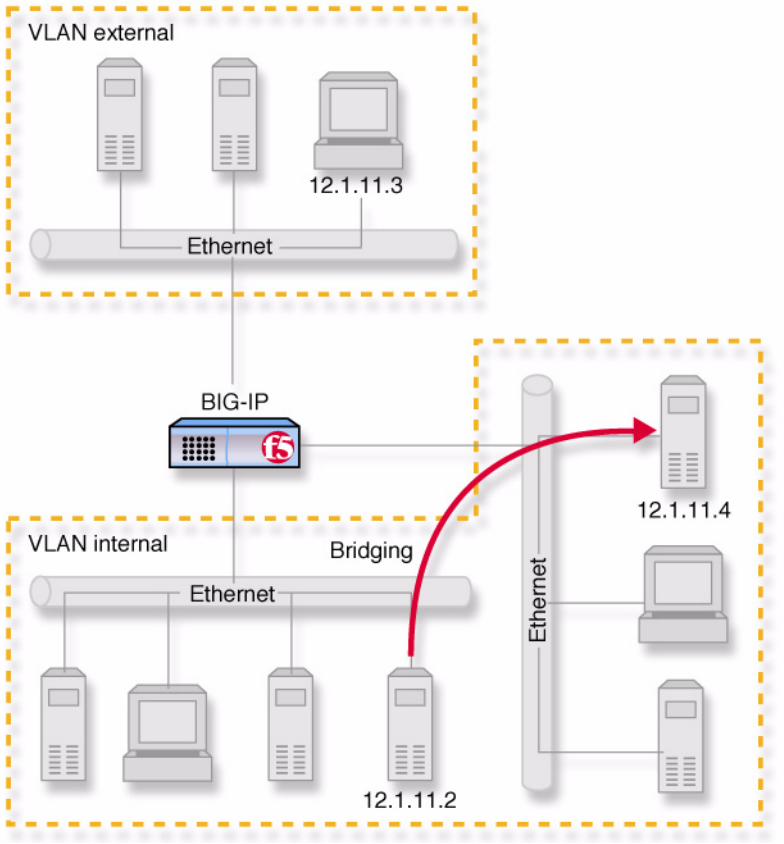

Figure 3.5 *Default VLAN configuration*

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 system 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 system 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 system 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, the BIG-IP system 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 the BIG-IP system, 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).

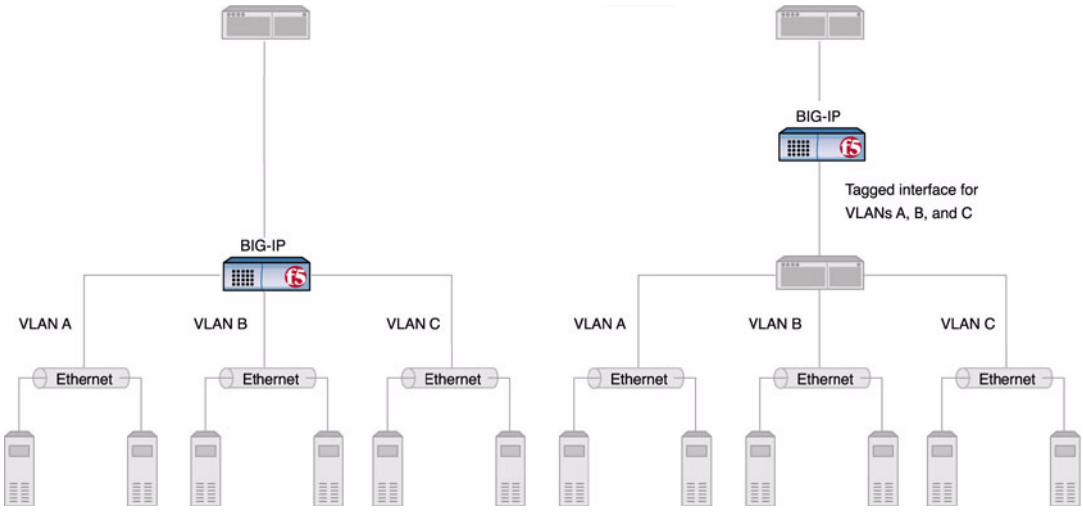

Figure 3.6 *Equivalent solutions using untagged and tagged interfaces*

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 system 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, the BIG-IP system 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 *Configuring VLAN groups*, on page 3-15. 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 system 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
```

Configuring 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.

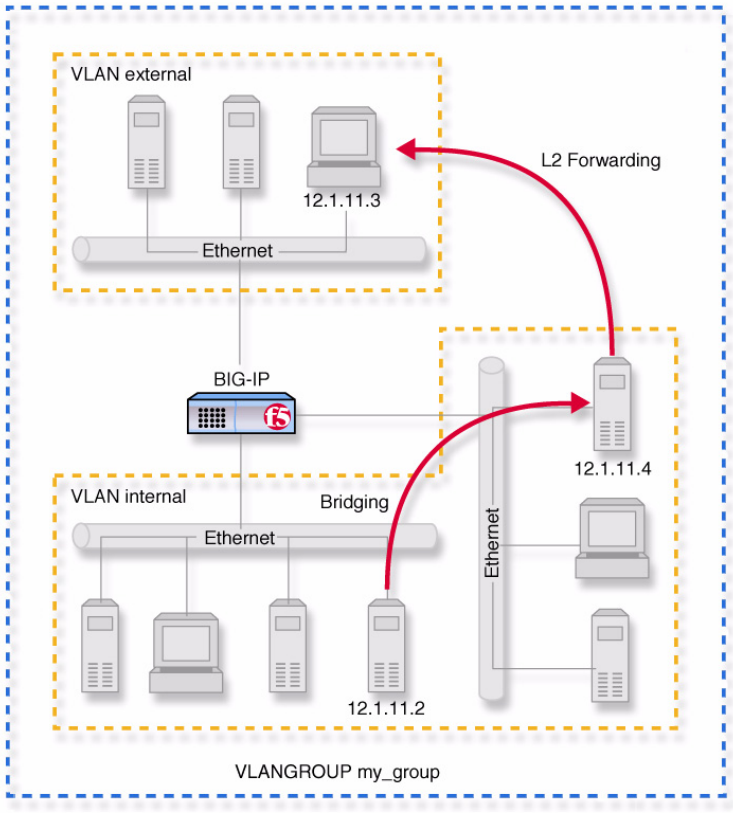

Figure 3.7 *VLANs and a VLAN group*

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 or change L2 forwarding (also known as **proxy forwarding**).

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 self** command. The syntax is as follows:

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

To verify that L2 forwarding is enabled

L2 forwarding is enabled for the VLAN group using the **VLAN proxy_forward** attribute. By default, this attribute is enabled when you create a VLAN group.

To verify that L2 forwarding is enabled, type the following command:

```
b vlangroup show
```

To change the operation of L2 forwarding

If you want to manage L2 forwarding for a specific VLAN group or groups, use the **bigpipe vlangroup** command. Enabling the **proxy_forward** attribute with this command results in a combination of L2 proxy ARP with L3 forwarding. You can either use this default type of L2 forwarding (default value = **0**), or change the value. Table 3.3 lists the allowed values.

Value	Description
0	The default L2 proxy ARP with L3 forwarding.
1	L2 forwarding with locally-unique bit, toggled in ARP response across VLANs.
2	L2 forwarding with the original MAC address of the remote system preserved across VLANs.

Table 3.3 *Allowed values for managing L2 forwarding on a VLAN group*

If you want to manage L2 forwarding globally for all VLAN groups, use the **bigpipe global** command as follows:

```
b global vlangroups [value]
```

Table 3.4 lists the allowed values.

Value	Description
opaque	A proxy ARP with layer 3 forwarding. The command line syntax for enabling this setting is: `b global vlangroups opaque`
translucent	Layer 2 forwarding with locally-unique bit, toggled in ARP response across VLANs. This is the default setting.
transparent	Layer 2 forwarding with the original MAC address of the remote system preserved across VLANs. The command-line syntax for enabling this setting is: `b global vlangroups transparent`

Table 3.4 *Allowed values for globally managing L2 forwarding*

To prevent L2 proxy ARP forwarding

In some cases, you might not want the active unit to forward proxy ARP requests to the standby unit, or to other hosts in the configuration. To exclude specific hosts from receiving forwarded proxy ARP requests, you can define a **proxy_arp_exclude** class that specifies the self IP addresses that you want to exclude. For example:

```
b class proxy_arp_exclude {host <self IP 1> host <self IP 2>  host <self IP N>}
```

To specify a value for downed links

You can specify the length of time that a BIG-IP unit in a VLAN group keeps its links down when they are dropped during a switch from active to standby mode. A BIG-IP unit drops its links so that any connected switches will recognize that all proxy ARPed MAC addresses are on the currently-active BIG-IP system and not on the standby unit.

The value is specified in tenths of seconds. Thus, a value of **50** is equivalent to 5 seconds. By default, this feature is disabled, with a value of **0**.

For example, the following command specifies a value of 5 seconds:

```
b global set standby_link_down_time = 50
```

Setting up security for VLANs

You can lock down a VLAN to prevent direct connection to the BIG-IP system 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, check 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 system. 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

Note

*For sensible operation, you must set the MAC masquerade address to be the same on both the active and standby units. To do this, configure the shared MAC address manually, by editing the **bigip_base.conf** file on both units. Do not use the **bigpipe config sync** command.*

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 system 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 system 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 system 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.*

Configuring VLAN mirroring

VLAN mirroring is an element of the Probe Control feature. Probe Control allows the BIG-IP system to replicate packets and send them to another network device. Typically, Probe Control is useful when deploying intrusion detection systems, which passively inspect all packets going through a network.

VLAN mirroring is similar to port mirroring, in that packets received by a VLAN are copied and sent to another VLAN or set of VLANs. In a VLAN mirroring configuration, this occurs for all traffic received on the source VLAN, regardless of the destination MAC address on the packet. Note that VLAN mirroring applies to Out-of-Band configurations only, which means that packets sent *from* the BIG-IP system, out through a specific VLAN, are not mirrored.

A VLAN configured to receive replicated packets is known as a ***mirror-target VLAN***.

Important

Once you have configured a mirror-target VLAN, you cannot use any IP addresses on that VLAN. Thus, you cannot use IP addresses such as virtual servers, SNATs, and self IP addresses on a VLAN configured for mirroring.

Figure 3.8 shows an example of entries in the **bigip.conf** file that enable VLAN mirroring.

```
vlan external {
    interfaces 1.1
    mirror vlans ids1 and ids2
    }
```

***Figure 3.8** Example of a VLAN mirroring configuration*

In the preceding example, the VLAN **external** replicates packets and sends them to VLANs **ids1** and **ids2**. VLANs **ids1** and **ids2** are the mirror-target VLANs.

To configure VLAN mirroring using the Configuration utility

1. In the navigation pane, click **Network**.
   The VLANs screen opens.
2. Click the **Add** button or click on an existing VLAN name to view its properties.
3. In the **Mirrored VLANs** box, select a VLAN name in the **Existing VLANs** box and move it to the **Mirrored VLANs** box, using the arrows (**>>**).
4. Click **Done** or **Apply**.

To configure VLAN mirroring from the command line

The following command syntax shows how to configure VLAN mirroring on two VLANs.

```
b vlan <vlan_name> mirror vlans <vlan1> <vlan2>
```

When using VLAN mirroring for load balancing, you can enable **hash** mode. The next section describes **hash** mode, followed by a description of VLAN mirroring as used by intrusion detection systems.

Hash mode

When you configure VLAN mirroring for **hash** mode, you can choose from two settings--**Enabled** or **Port Enabled**.

Enabled

When **hash** mode is set to **Enabled**, the replicated packets are not sent to every VLAN in the mirror list. Instead, the BIG-IP system hashes the IP addresses on the packet and sends the packet to only one of the mirrored VLANs, based on the computed hash.

Figure 3.9 shows the same mirrored VLAN configuration as above, with **hash** mode set to **Enabled**.

```
vlan external {
    interfaces 1.1
    mirror vlans ids1 and ids2
    mirror hash enable
    }
```

Figure 3.9 *Example of hash mode set to Enabled*

To set hash mode to Enabled using the Configuration utility

1. In the navigation pane, click **Network**.
   The VLANs screen opens.
2. Click the **Add** button or click on an existing VLAN name to view its properties.
3. In the **Mirror Hash** box, choose **Enabled**.
4. Click **Done** or **Apply**.

To set hash mode to Enabled from the command line

The following command syntax shows how to set hash mode to **Enabled** when you configure VLAN mirroring.

```
b vlan <vlan_name> mirror vlans <vlan1> <vlan2> hash enable
```

Port Enabled

Similar to the **Enabled** setting, the **Port Enabled** setting causes TCP or UDP ports to be included in the computed hash.

Figure 3.10 shows the same mirrored VLAN configuration as above, with **hash** mode set to **Port Enabled**.

```
vlan external {
    interfaces 1.1
    mirror vlans ids1 and ids2
    mirror hash port enable
    }
```

Figure 3.10 *Example of hash mode set to Port Enabled*

◆ Note

*Setting the hash mode to **Port Enabled** can increase the granularity of load balancing, However, this mode is incompatible with IP fragmentation. If your network traffic includes IP fragments, it is recommended that you set the hash mode to **Enabled**.*

To set hash mode to Port Enabled using the Configuration utility

1. In the navigation pane, click **Network**.
   The VLANs screen opens.
2. Click the **Add** button or click on an existing VLAN name to view its properties.
3. In the **Mirror Hash** box, choose **Port Enabled**.
4. Click **Done** or **Apply**.

To set hash mode to Port Enabled from the command line

The following command syntax shows how to set hash mode to **Port Enabled** when you configure VLAN mirroring.

```
b vlan <vlan_name> mirror vlans <vlan1> <vlan2> hash port enable
```

Handling traffic from Intrusion Detection Systems

Typically, the VLAN mirroring feature is used to send packets to a passive intrusion detection system (IDS). An IDS inspects packets and looks for threatening content.

If the IDS detects a packet to be part of a network attack, the IDS might attempt to send one or more TCP resets to the client and server as a way to terminate the connection. When this happens, it is a mirror-target VLAN that receives the packets instead of the original source VLAN.

By default, once the mirror-target VLAN receives the packets, it forwards the packets to the source VLAN (in our example, VLAN **external**). You can disable this behavior by resetting the **mirror_vlan_forwarding** variable, using the **bigpipe global** command. Disabling this variable causes any packets received on a mirror-target VLAN to be discarded.

To disable forwarding of packets to a source VLAN using the Configuration utility

1. In the navigation pane, click **System**.
2. Click the Advanced Properties tab.
3. In the **mirror_vlan_forwarding** box, click on the check box to remove the check.
4. Click **Apply**.

To disable forwarding of packets to a source VLAN from the command line

The command syntax for disabling packet forwarding in a mirrored configuration is as follows:

```
b global mirror_vlan_forwarding disable
```

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 system. 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 system. (A floating self IP address is the address to which the servers behind the BIG-IP system 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 Configuration utility or the **bigpipe 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 system:

```
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, see Chapter 10, *Address Translation: SNATs, NATs, and IP Forwarding*.

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 3.11.

```
# 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 3.11 *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 system 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.*

Managing the SSH Console

An SSH console gives you the ability to use a command line interface to securely manage your local BIG-IP system. You can either use the MindTerm SSH console that is available through the Configuration utility, or you can download a different SSH console, using the initial screen of the BIG-IP browser interface.

Using the MindTerm SSH Console

With the MindTerm SSH Console, you can open an SSH session for the BIG-IP system 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 system as you normally would.

◆ Note

When you use the MindTerm SSH Console, you can only administer the local BIG-IP system. 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 an 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 system, 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 system does not support encrypted connections, you can use a Telnet shell for remote command line access.

Downloading the SSH client from the web server

1. Connect to the BIG-IP system using **https://** rather than **http://** in the URL.
2. In the Additional Software Downloads section, click the **SSH Clients** link.
3. From the SSH Clients page, you can choose the SSH Client appropriate to your operating system.

Setting up an 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 system 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 an 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 system:

   ```
   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 system 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 system. You also must set up routes for the nodes that the BIG-IP system manages. You may also want to configure the BIG-IP so that dynamic routing information can automatically be updated on the BIG-IP system.
- **Configuring DNS on the BIG-IP system**
  You may need to configure the BIG-IP system 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 system**
  There are some special requirements that you need to take into account when configuring email on the BIG-IP system.

Addressing routing issues

The BIG-IP system 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 system into production.

The BIG-IP system supports static route configurations, dynamic routing (by way of BGP4, RIP1, RIP2, and OSPF), and subnetting. However, the BIG-IP system is also designed to eliminate the need for you to modify routing tables on a router that routes to a BIG-IP system. Instead, the BIG-IP system 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 system to a gateway to the external network
- Routing from content servers to the BIG-IP system
- Routing between a BIG-IP system 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 system to a gateway to the external network

The BIG-IP system needs a route to the external network. For most configurations, this should be configured as the default gateway pool on the BIG-IP system.

During installation, you were prompted to configure a default route for the BIG-IP system. 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 system

The content servers being load balanced by the BIG-IP system need to have a default route set to the internal shared floating IP alias of the BIG-IP system. 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 system and content servers on different logical networks

If you need to configure the BIG-IP system to use one or more nodes that actually sit on a different logical network from the BIG-IP system, 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 system internal interface.

In the following examples, the nodes are on **192.168.6.0/24** and the BIG-IP system 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 system, you 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 this syntax, where **<ip addr>** is the IP address on the internal interface:

```
route add -net 192.168.6 -interface <ip addr>
```

Note

*Make sure that you have defined the interface correctly in the **/etc/hosts** file.*

Case 2: Different LANs

If you have nodes on different LANs from the BIG-IP system, you need to add a static gateway route on the BIG-IP system 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 system:

```
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 system 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 system. 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 system, as long as the default path eventually routes through the BIG-IP system.

Setting up dynamic routing with the Advanced Routing Modules

You can configure dynamic routing using Advanced Routing Modules (ARMs). ARMs correspond to the following protocols or modules: Border Gateway Protocol (BGP), network services module, Open Shortest Path First (OSPF) Protocol, and Router Information Protocol (RIP).

For information on setting up dynamic routing using the ARMs, see the *Command Reference* guide that corresponds to the appropriate module. Available guides are:

- *BGP Command Reference*
- *NSM Command Reference*
- *OSPF Command Reference*
- *RIP Command Reference*

Configuring static routes in /config/routes

You can create the file **/config/routes** on the BIG-IP system for configuring static route information. The information you add to **/config/routes** is synchronized between units in a BIG-IP redundant system. 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 3.12.

```
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 3.12 *Example entries in* ***/config/routes***

Configuring DNS on the BIG-IP system

If you plan to use DNS in your network, you can configure DNS on the BIG-IP system. There are three different DNS issues that you may need to address when setting up the BIG-IP system:

- Configuring DNS resolution on the BIG-IP system
- 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 system, you can use the address, host name, or fully qualified domain name (FQDN).

The BIG-IP system 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 system 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 3.13.

```
; 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 3.13 *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 3.14, for an example of how to make the entry in the **/etc/irs.conf** file.

```
hosts           local   continue
hosts           dns
```

Figure 3.14 *Sample entry for the* ***/etc/irs.conf*** *file*

Configuring DNS proxy

The BIG-IP system 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 system. 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 system 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 system. 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 3.15.

```
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 3.15 *Sample zone table with two Web sites and four servers*

In the BIG-IP system 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 3.16.

```
www.SiteOne.com  IN A 192.168.100.231
www.SiteTwo.com  IN A 192.168.100.232
```

Figure 3.16 *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 system is email. You can configure the BIG-IP system to send email notifications to you, or to other administrators. The BIG-IP system uses Sendmail as its mail transfer agent. The BIG-IP system 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 system 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. 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 this command:

   ```
   /usr/sbin/sendmail -bd -q30m
   ```

Using a serial terminal with the BIG-IP system

There are a couple of different ways to add a serial terminal to the BIG-IP system. 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 system

Connect a serial line cable between the terminal device and the BIG-IP system. On the back of BIG-IP system 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 system. A null modem cable is required.

The connector is wired as a DTE device, and uses the signals described in Table 3.5.

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 3.5 *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 system. 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 3.5.

◆ Note

You can achieve acceptable operation by wiring pins 7 to 8 and pins 1 to 4 at the back of the BIG-IP system (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 system 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 system.
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 system.

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 system.
2. Connect the serial terminal to the BIG-IP system. When there is no keyboard connected to the BIG-IP system, the BIG-IP system 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 system.

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 system 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 system.

4. Turn on the serial terminal.
5. Reboot the BIG-IP system.

Once you configure a serial terminal as the console for the BIG-IP system, 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.*

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 3-3.

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 system. 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 system 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 system, 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** daemon does not automatically restart when you synchronize configurations between units in a BIG-IP redundant system. 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
```

PART III
THE HIGH-LEVEL NETWORK

4

Pools

- Introducing pools
- Managing pools
- Load balancing methods
- Setting persistence
- Redirecting HTTP requests
- Inserting and erasing HTTP headers
- Configuring the Quality of Service (QoS) level
- Configuring the Type of Service (ToS) level
- Disabling SNAT and NAT connections
- Enabling a forwarding pool
- Configuring a clone pool

Introducing pools

A load balancing pool is the primary object in the high-level network. 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.

A ***pool*** is a set of devices grouped together to receive traffic according to a load balancing method. Pools are associated with a specific virtual server. Thus, traffic coming into a virtual server is normally directed to one of the associated pools. As an option, you can redirect traffic to a different pool using the BIG-IP system's iRules feature. For more information on iRules, see Chapter 5, *iRules*.

Once a pool receives traffic, either directly from a virtual server or through a rule, the pool can optionally perform a number of different operations, such as inserting a header into an HTTP request, setting the Quality of Service or Type of Service level within a packet, or redirecting a request to a fallback destination.

Perhaps the most useful and flexible feature with respect to pools, however, is the BIG-IP system's Universal Inspection Engine (UIE). The ***UIE*** allows a pool to make load-balancing decisions based on information contained either in headers, or in the content of a packet. Thus, a pool can perform load-balancing operations such as sending traffic to a specific node within the pool, or enabling persistence based on any string or node that you define.

Required pool attributes

When creating a basic pool, you must specify two pool attributes. Table 4.1 lists and describes these required attributes.

Required Attribute	Description
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 IP address	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.

Table 4.1 *Required pool attributes*

◆ Note

A third attribute, the load balancing method, is required; however, the BIG-IP system automatically specifies a default mode when you create the pool. For more information, see ***Load balancing methods****, on page 4-6.*

Optional pool attributes

You create a pool using the Configuration utility or the **bigpipe pool** command. When you create the pool, you assign a pool name and specify a number of attributes, such as the members of the pool, the load balancing method you want the BIG-IP system to use to select pool members, and the type of persistence required, if any. You can also configure the pool to send certain types of traffic to a specific member of that pool.

When creating a pool, you can configure several optional pool attributes. Table 4.2 lists the optional attributes you can configure for a pool.

Pool Attribute	Description
Load balancing method	You can define a specific load balancing method for a pool, and you can configure priority-based member activation. Various pools can be configured with different load balancing methods.
Persistence method	You can assign a pre-defined persistence type to a pool, or you can specify an expression that enables persistence on any string. You can also specify an expression that directly selects a node within a pool.
HTTP redirection	You can redirect HTTP requests to a fallback host, protocol, port, or URI path.
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.
HTTP header erase	You configure a pool to erase the contents of a header in an HTTP request. This is useful when headers contain sensitive information that you do not want to forward over a network.
Quality of Service (QoS) level	You can configure a pool to set a specific QoS level within a packet, based on the targeted pool.
Type of Service (ToS) level	You can configure a pool to set a specific ToS level within a packet, based on the targeted pool.
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.
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.
Mirroring	The **mirror** attribute mirrors a persistence record over to a standby unit. A persistence record identifies the connections to be persisted.
Clone pool	The clone pool attribute is used for intrusion detection. When the attribute is enabled, a regular pool has a clone pool configured for it.

Table 4.2 *Optional pool attributes*

A full description of these attributes begins with the section *Load balancing methods*, on page 4-6.

Managing 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.

Creating a pool

You can create either a basic pool, specifying just a pool name and pool members, or you can create a pool specifying multiple attributes, such as persistence types, header insertion, and Quality of Service/Type of Service levels.

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 type the following command:

```
b pool my_pool { member 11.12.1.101:80 member 11.12.1.100:80
}Table 4.3 lists all pool attributes and shows the syntax that
you use to configure them with the bigpipe pool command.
```

Pool Attribute	Syntax
Pool name	A string from 1 to 31 characters, for example: **new_pool**
Member definition	**member <ip_address>:<service> [ratio <value>] [priority <value>]**
load balancing method	**lb_method [rr | ratio | fastest | least_conn | predictive | observed | ratio_member | fastest_member | least_conn_member | observed_member | predictive_member | dynamic_ratio]**

Table 4.3 *Syntax for pool attributes*

Pool Attribute	Syntax
persistence type	**persist [simple | cookie | ssl | sip | sticky | msrdp | <expression>]** **persist_timeout [<timeout_value>]** **select [<expression>]**
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 and erase	**header_insert <quoted string>** **header_erase <quoted string>**
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**
clone pool	**clone before | clone after <pool_name>**

Table 4.3 *Syntax for pool attributes*

Modifying a pool

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.

To modify the attributes of a pool, you can use either the Configuration utility or the **bigpipe pool** command.

To modify a pool using the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the **Pool Name** list, click on the name of the pool that you want to modify.
   This displays the properties of that pool.

3. Change any of the pool attributes shown.
4. Click **Apply**.

To modify a pool from the command line

The following example shows how to change the default load-balancing mode from Round Robin to Predictive and add two new members to the pool:

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

Deleting a pool

You can delete an existing pool, using either the Configuration utility or the **bigpipe pool** command.

To delete a pool using the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the **Pool Name** list, locate the pool that you want to delete and click the Delete button.
   A confirmation dialog box appears.
3. Click **OK**.

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.

Displaying a pool

Displaying a pool means looking at the properties of that pool. You can look at the properties of an existing pool using either the Configuration utility or the **bigpipe pool** command.

To display a pool using the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the **Pool Name** list, click on the name of the pool that you want to modify.
   This displays the properties of that pool.

To display one or more pools from the command line

Use the following command to display all pools:

```
b pool show
```

Use the following command 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.

Load balancing methods

Load balancing is an integral part of the BIG-IP system. Configuring load balancing on the BIG-IP system 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.

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

The default load balancing method on the BIG-IP system is Round Robin, which simply passes each new connection request to the next server in line. All other load balancing methods 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, described in this section.

Round Robin

This is the default load balancing method. 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

The BIG-IP system 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 method, 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 method, 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 load balancing traffic to RealNetworks® RealSystem® Server 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 implement Dynamic Ratio load balancing, you must first install and configure the necessary server software for these systems. For more information, see *Configuring Dynamic Ratio load balancing*, on page 4-11.

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 system 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 method, 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 method, 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 system 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 method, 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 method for a pool

A load balancing method 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 *Managing 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 method using the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. The next action depends on your intent:
   - 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 name in the Pool Name list.
     The Pool Properties screen opens.
3. 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 method 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 navigation pane, click **Pools**.
   The Pools screen opens.
1. In the Pool Name list, click a pool name.
   This displays the properties of that pool.
2. In the **Current Members** list, click the member you want to edit.
3. Click the Back button (**<<**) to pull the member into the resources section.
4. Change or add the ratio value for the member.

5. Click the Add button (**>>**) to add the member to the **Current Members** list.
6. 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 method 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}
```

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

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 method, 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 method has the same effect as the Round Robin load balancing method.

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 one or more members using the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the Pool Name list, click the name of the pool.
   This displays the properties of that pool.
3. In the **Current Members** list, view the **r** number following each IP address.

To display the ratio weights for one or more node addresses from the command line

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*

The following command syntax displays the ratio setting for one or more node addresses:

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

Configuring Dynamic Ratio load balancing

You can configure Dynamic Ratio load balancing for pools that consist of RealNetworks® RealServer servers, Windows servers 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 implement Dynamic Ratio load balancing, the BIG-IP system provides a special monitor plugin file and a health monitor template for each type of server (except for a server equipped with an SNMP agent. In this case, the BIG-IP system needs to provide the monitor template only).

For each server, the monitor plugin must be installed on the server, while the monitor template is used to create a monitor that resides on the BIG-IP system. Once created, the monitor communicates directly with the server plugin.

Table 4.4 shows, for each server type, the required monitor plugin and the corresponding health monitor templates.

Server Type	Monitor Plugin	BIG-IP Monitor Template
RealServer Windows server RealServer UNIX server	**F5RealMon.dll** **f5realmon.so**	real_server
Windows server with WMI	**F5Isapi.dll**	wmi
Windows 2000 Server server UNIX server	**SNMP agent** **UC Davis SNMP agent**	snmp_dca and snmp_dca_base

Table 4.4 *Monitor plugins and corresponding monitor templates*

Configuring RealSystem Server systems

For RealSystem Server systems, the BIG-IP system provides a monitor plugin that gathers the necessary metrics when the plugin is installed on the RealSystem Server system. Configuring a RealSystem Server for Dynamic Ratio load balancing consists of four tasks:

- Installing the monitor plugin on the RealSystem server
- Configuring a **real_server** health check monitor on the BIG-IP system
- 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 a RealSystem Server system (Windows version)

Use the following procedure to install the monitor plugin on a Windows RealSystem Server system.

1. Download the monitor plugin **F5RealMon.dll** from the BIG-IP system. 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 RealSystem Server process is running, restart it.

To install the monitor plugin on a RealSystem Server system (UNIX version)

1. From the BIG-IP CD, copy the file **F5RealMonsrc.tgz** from the directory **downloads/rsplugins**.
2. Download the RealSystem Server SDK from <<http://proforma.real.com/rnforms/resources/server/realsystemsdk/index.html>>
3. Use the UNIX utilities **gunzip** and **tar** to uncompress the file **F5RealMonsrc.tgz**, as follows:

   ```
   gunzip F5ReaMonsrc.tgz
   tar -xpf F5RealMonsrc.tar
   ```

   A list of files appears that includes two makefiles, **linux-2.0-libc6-i386.mak** and **sunos-5.7-sparc.mak**, for Linux and SunOS systems respectively.
4. In both makefiles, change the paths that point to the SDK location.
5. Type one of the following commands, depending on your operating system:

   ```
   make -f  linux-2.0-libc6-i386.mak (for Linux)
   make -f sunos-5.7-sparc.mak (for Sunos)
   ```

   This creates a source file for the RealSystem Server plugin, **F5RealMon.so**.
6. Move the **F5RealMon.so** plugin to the **Plugins/** directory under the RealServer home directory.
7. Start RealSystem Server.

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 server-specific metrics and metric attribute default values is shown in Table 4.5.

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 4.5 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 4.5 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 running Windows Management Instrumentation (WMI), the BIG-IP system 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 system
- 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 system. 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**.
    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 user name 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 4.6.

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 4.6 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 4.6 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 4.6 wmi** monitor commands and metrics*

For more information about the metric coefficients and thresholds, refer to the description accompanying Table 4.5, *real_server monitor metrics*, on page 4-14. 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 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>...}
```

Configuring an SNMP agent

The BIG-IP system 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

The BIG-IP system 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 Chapter 11, Monitors.

Figure 4.4 shows a monitor based on the **snmp_dca** monitor template. This monitor uses the default metric values. You 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 ".1.3.6.1.4.1.2021.4.6.0"
    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. The values shown are default values, except for the value of USEROID, which is user-defined.

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

    USEROID ".1.3.6.1.4.1.2021.4.6.0"
    USEROID_COEFFICIENT "1.0"
    USEROID_THRESHOLD "90"
    }'
```

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

◆ Note

Note that in the preceding 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 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, the BIG-IP system 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 system 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 system 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 system 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 system again limits traffic to that group.

Setting persistence

If you are setting up an e-commerce or other type of transaction-oriented site, you might need to configure persistence on the BIG-IP system. Persistence is one of the pool attributes listed in Table 4.2, on page 4-2.

Whether you need to configure persistence or not depends on how you store client-specific information, such as items in a shopping cart or airline ticket reservations. For example, you might 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 enable persistence, returning clients can bypass load balancing and instead can go to the node where they last connected in order to access their saved information.

The BIG-IP system 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 assigned to the pool.

You enable persistence by assigning a persistence attribute to a pool, either through the Configuration utility or the **bigpipe pool** command. Two attributes are available for you to assign--the **persist** attribute (formerly known as the **persist_mode** attribute) and the **select** attribute.

Note

*Because the **persist_mode** attribute is backward-compatible, there is no need to change existing instances of the attribute name from **persist_mode** to **persist** in the **bigip.conf** file.*

Persistence types

You can choose from several types of persistence when configuring a pool. The first type listed below, universal persistence, is the most flexible type of persistence. By including an expression in your pool definition, you can persistence on anything within the header or the content of a packet.

The other persistence types are pre-defined, meaning that the element on which the persistence is based cannot be changed.

The persistence types are:

- **Universal persistence**
  Universal persistence allows you to write an expression that defines what to persist on in a packet. The expression, written using the same expression syntax that you use in rules, defines some sequence of bytes to use as a session identifier.
- **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 system 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 connections between clients and servers in WTS client-server configurations.

Note

All persistence methods are attributes of pools.

Table 4.7 lists the types of persistence, the attribute you assign to enable that persistence type, and the value you assign to the attribute.

Persistence Type	Required Attribute	Attribute Value
Universal	**persist** or **select**	**<expression>**
Simple	**persist**	**simple**
HTTP cookie	**persist**	**cookie**
SSL	**persist**	**ssl**
SIP Call-ID	**persist**	**sip**
Destination Address Affinity (sticky)	**persist**	**sticky**
WTS	**persist**	**msrdp**

Table 4.7 *Required attributes and values for enabling persistence*

The following sections describe the various types of persistence.

Universal persistence

Universal persistence is based on a user-written expression within a pool definition that defines some sequence of bytes to use as a session identifier. Expressions defined for this type of persistence must be either a single function or variable name, or enclosed in parentheses. (For the complete syntax of expressions, see Chapter 5, *iRules*.)

Universal persistence allows you to persist on anything within the header or the content of a packet, or to directly specify the node (pool member) to which the pool should send a packet. The following sections describe these persistence types.

Directing traffic based on any data in the connection

By assigning the **persist** attribute to a pool, you can enable persistence based on any data that you specify within an expression. The expression can include one or more of the following functions.

Note

Because these functions are part of the basic rules syntax, the parameters for these functions, and examples of their use, are fully described in Chapter 5, iRules.

- **findstr()** - Finds a string within another string, and returns the string starting at the offset specified from the match.
- **substr()** - Returns the string starting at the offset specified.
- **getfield()** - Splits a string on a character, and returns the string corresponding to the specific field.
- **findclass()** - Finds the member of a class that contains the result of the specified expression, and returns that class member.
- **decode_uri()** - Evaluates the expression and returns a string with any **%XX** escape sequences decoded as per HTTP escape sequences defined in RFC2396.
- **domain()** - Parses and returns up to the specified number of trailing parts of a domain name from the specified expression.
- **imid()** - Parses the **http_uri** variable for an i-mode identifier string that can be used for i-mode persistence.

An example of this type of persistence is i-mode persistence. ***i-mode*** is a service that gives mobile phone users wireless Web browsing and email capabilities. ***i-mode persistence*** is implemented using a function that allows the BIG-IP system to perform persistence on an i-mode session identifier in the query of string of a URI. Using this persistence type eliminates the need to establish a predefined relationship between a session identifier and a pool.

Important

Two different search strings could result in the two return strings being hashed to the same key, and stored as one key in the hash table. If this happens, one of the two requests could persist to the wrong pool member. To avoid this problem, make sure that you write expressions that return unique string values.

Figure 4.7 shows an example of **/etc/bigip.conf** file entries for persistence based on expressions used as hashes for pool members. Note that this example includes an entry for i-mode persistence.

```
pool my_pool {
   lb_mode fastest
   min_active_members 2

   persist domain(http_host, 2)
    persist_timeout 120
    member 10.12.10.1:80
    member 10.12.10.3:80

   persist findstr(http_uri, "user=", 5, '&')
    persist_timeout 30
    member 10.12.10.1:80
   member 10.12.10.3:80

   persist (domain(http_host, 2) + http_header("User-Agent"))
    persist_timeout 60
    member 10.12.10.1:80
    member 10.12.10.3:80

   persist imid
    persist_timeout 120
    member 10.12.10.1:80
    member 10.12.10.3:80
}
```

Figure 4.7 *A pool definition that enables persistence based on data found in the connection*

To enable persistence for pools based on any data in a connection 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 **Expression** 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.)
   - **Expression**
     Using the allowed rule functions, type an expression that defines the data on which to persist in the connection.
5. Click the **Apply** button.

To enable persistence based on any data in a connection from the command line

To enable a pool to direct traffic to a specific node, use the following command line syntax:

```
b pool <pool_name> modify ( persist <expr> persist_timeout
<timeout> )
```

For detailed descriptions of the functions that specify any data in a connection, see Chapter 5, *iRules*.

Directing traffic to a specific node

By assigning the **select** attribute to a pool, you can specify an expression that returns the node address of a pool member. This type of expression includes one of the following functions.

Note

Because these functions are part of the basic rules syntax, the parameters for these functions, and examples of their use, are fully described in Chapter 5, iRules.

- **node()** - Returns a literal node address converted from either a string representation of an address and port or a literal number representing the node address as an integer.
- **mapclass2node** - Represents a short-hand combination of the functions **findclass()**, **findstr()**, and **node()**.
- **wlnode()** - Returns a literal node address converted from either a string representation of an address and port, a literal number representing the node address as an integer, or a literal node address.

Figure 4.8 shows an example of the **/etc/bigip.conf** file entries that show expressions that select specific node addresses for persistence.

```
pool my_pool {
    lb_mode fastest
    min_active_members 2

   select wlnode(http_cookie("JSESSIONID"))
    member 10.12.10.1:80
    member 10.12.10.3:80

   select node("10.0.0.3:80")
   member 10.12.10.1:80
   member 10.12.10.3:80

   select node(getfield(http_cookie("SERVER"), ';', 1))
   member 10.12.10.1:80
   member 10.12.10.3:80
}
```

Figure 4.8 *A pool definition that enables persistence by selecting a specific node*

To enable persistence by selecting a specific node using the Configuration utility

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the **Pool Name** list, click the pool name for which you want to configure simple persistence.
   This displays the properties of that pool.
3. Click the Persistence tab.
   The Persistence Properties screen opens.
4. In the **Node Select Expression** box, type an expression that defines a specific node.
5. Click the **Apply** button.

To enable persistence by selecting a specific node from the command line

To enable a pool to direct traffic to a specific node, use the following command line syntax:

```
b pool <pool_name> modify ( select <expr> )
```

For detailed descriptions of the functions that directly select a node, see Chapter 5, *iRules*.

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 system checks to see if that client previously connected, and if so, returns the client to the same node.

You might 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 system 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 system 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. In the **Pool Name** list, click the pool name for which you want to configure simple persistence.
   This displays the properties of that pool.

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 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 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 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 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 system 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 system 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 system.

To activate Insert mode using the Configuration utility

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

2. In the **Pool Name** list, click the pool name for which you want to set up Insert mode.
   This displays the properties of that pool.
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 cookie cookie_mode insert \
    cookie_expiration <timeout> <member definition> }
```

The **<timeout>** value for the cookie is written as **<days>d hh:mm:ss**.

Rewrite mode

If you specify Rewrite mode, the BIG-IP system 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 system 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 Pool Name list, click the pool name for which you want to set up Rewrite mode.
   This displays the properties of that pool.
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 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 system 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 system.

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the Pools Name list, click the pool name for which you want to set up Passive mode.
   This displays the properties of that pool.
3. Click the Persistence tab.
   The Persistence screen opens.
4. Select **Passive HTTP Cookie** mode.
5. Click the **Apply** button.

To activate Passive mode cookie persistence from the command line

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

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

Note

Passive mode does not require a ***<timeout>*** *value.*

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 system 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 system 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 **Pool Name** list, click the pool name for which you want to set up hash mode persistence.
   This displays the properties of that pool.
3. Click the Persistence tab.
   The Persistence screen opens.
4. Click the **Cookie Hash** button.
   Set the following values (see Table 4.8 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 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 4.8.

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 4.8 *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 system 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 system 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 system 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 system stores a given SSL session ID before removing it from the system.

◆ Note

Do not use the SSL persistence attribute to enable persistence on pools that load balance traffic resulting from SSL proxies on which SSL termination is enabled. To enable persistence for connections terminated by an SSL proxy, see Chapter 7, SSL Accelerator Proxies.

To activate SSL persistence from the command line

1. In the navigation pane, click **Pools**.
   The Pools screen opens.
2. In the Pool Name list, click the appropriate pool name.
   This displays the properties of that pool.
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 system 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 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 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

The BIG-IP system 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 system 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 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 Chapter 6, *Virtual Servers*. Follow these steps to configure destination address affinity.

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

2. In the **Pool Name** list, click the pool name for which you want to set up destination address affinity.
   This displays the properties of that pool.
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 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

This release includes an updated version of the BIG-IP system Windows Terminal Server (WTS) persistence feature. WTS persistence provides an efficient way of load balancing traffic and maintaining persistent connections between Windows® clients and servers that are running the Microsoft® Terminal Services service. The recommended scenario for enabling the BIG-IP system WTS persistence feature is to create a load balancing pool that consists of members running Windows .NET Server 2003, Enterprise Edition, where all members belong to a Windows cluster and participate in a Windows session directory.

◆ Note

Servers running Windows .NET Server 2003, Enterprise Edition, with the Terminal Services service enabled, are referred to in this chapter as ***Terminal Servers****.*

Benefits of WTS persistence

Without WTS persistence, Windows Terminal Servers, when participating in a session directory, map clients to their appropriate servers, using redirection when necessary. If a client connects to the wrong server in the cluster, the targeted server checks its client-server mapping and performs a redirection to the correct server.

When BIG-IP system WTS persistence is enabled, however, a Windows Terminal Server participating in a session directory always redirects the connection to the same BIG-IP virtual server, instead of to another server

directly. The BIG-IP system then sends the connection to the correct Windows Terminal Server. Also, when WTS persistence is enabled on a BIG-IP system and the servers in the pool participate in a session directory, the BIG-IP system load balances a Terminal Services connection according to the way that the user has configured the BIG-IP system for load balancing. Thus, the use of Windows Terminal Servers and the Session Directory service, combined with the BIG-IP WTS persistence feature, provides more sophisticated load balancing and more reliable reconnection when servers become disconnected.

Server Platform issues

By default, the BIG-IP system with WTS persistence enabled load balances connections according to the way that the user has configured the BIG-IP system for load balancing, as long as Session Directory is configured on each server in the pool. Because Session Directory is a new feature that is only available on Windows .NET Server 2003, Enterprise Edition platforms, each server in the pool must therefore be a Windows .NET Server 2003, Enterprise Edition server if you want to use WTS persistence in default mode. Also, each client system must be running the remote desktop client software that is included with any .NET Enterprise server or Windows XP system.

If, however, you want to enable WTS persistence but your server platforms are running older versions of Windows (on which Session Directory is not available), you can enable WTS persistence in non-default mode. This causes the BIG-IP system to connect a client to the same Windows server by way of the user name that the client provides. You can enable WTS persistence in this way by setting a global variable on the BIG-IP system, called **msrdp no_session_dir**, which disables Session Directory on any pool created with the **msrdp** attribute. Note that enabling WTS persistence in non-default mode (that is, with no Session Directory available on the servers) is less preferable than the default mode, because it provides limited load-balancing and redirection capabilities.

The following sections describe how to enable WTS persistence with and without Windows Session Directory.

Configuring WTS persistence with Session Directory

To enable WTS persistence in the default mode, you must configure Session Directory on each Windows Terminal Server in your load balancing pool. In addition to configuring Session Directory, you must perform other Windows configuration tasks on those servers. However, before you configure your Windows Terminal Servers, you must configure your BIG-IP system, by performing tasks such as creating a load-balancing pool and designating your Windows Terminal Servers as members of that pool.

The following two sections describe the BIG-IP system configuration tasks that are required to enable WTS persistence in default mode for a Windows client-sever configuration running Windows Terminal Services.

Configuring WTS persistence on the BIG-IP system

To configure WTS persistence on the BIG-IP system, you must perform the following tasks.

- Enable TCP service **3389**
- Create a pool with WTS persistence enabled
- Create a virtual server

The following sections describe these tasks.

To enable TCP service 3389

To enable TCP service **3389**, use the following command:

```
b service 3389 tcp enable
```

Optionally, you can map this port from **3389** to **443** in order to allow traffic to pass more easily through a firewall.

To create a pool with WTS persistence enabled

To create a pool that is configured for WTS persistence, use the **bigpipe pool** command, as in the following example. Remember that the pool members must already be members of a Windows cluster.

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

To create a virtual server

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
```

Configuring WTS persistence without Session Directory

When a server has no Session Directory, the server cannot share sessions with other servers, and therefore cannot perform any redirections when a connection to a server becomes disconnected. In lieu of session sharing, Windows clients provide data, in the form of a user name, to the BIG-IP system to allow the BIG-IP system to consistently connect that client to the same server. Enabling WTS persistence to behave in this way is the non-default mode.

To configure WTS persistence when the servers do not have Session Directory

1. Perform the BIG-IP system configuration tasks that are described in *Configuring WTS persistence on the BIG-IP system*, on page 4-42.
2. Set a BIG-IP system global variable, **msrdp no_session_dir**. Setting this global variable disables Session Directory on all pools on which the **msrdp** attribute is set. To set the **msrdp no_session_dir** global variable, use the following command-line syntax:

   ```
   b global msrdp no_session_dir enable
   ```

3. Verify that the Terminal Services service is running on each Windows server in your load-balancing pool.

Note

*To implement WTS persistence, you must also configure your Windows Terminal Servers. For more information, see the **BIG-IP Solutions Guide**.*

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 system 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 system to maintain persistence for all connections requested by the same client, regardless of which virtual server hosts each individual connection initiated by the client.

The following two sections describe these two persistence options.

Maintaining persistence across virtual servers that use the same virtual addresses

When this mode is turned on, the BIG-IP system 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 method defined for the pool.

A BIG-IP system configuration could include 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 system uses the persistence session established with the first connection to determine the node that should receive the connection request, rather than the load balancing method. The BIG-IP system 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 system starts tracking a new persistence session, and it uses the load balancing method 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 system 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 system 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 method.

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
```

For example, suppose that a client makes an initial connection to **v1:http**, and the BIG-IP system load balancing mechanism chooses **n1:http** as the node. If the same client subsequently connects to **v1:ssl**, the BIG-IP system 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 system 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
```

Redirecting HTTP requests

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 system. 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 4.9 redirects the request to **http://redirector.siterequest.com**.

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

Figure 4.9 *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 4.9 shows how different fallback host specifications are resolved.

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

Table 4.9 *Resolutions for fallback host specifications*

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 Chapter 5, *iRules*.)

Table 4.10 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 4.10 *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 4.11 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.siterequest.com:8080/sample
%h/unavailable	change path, remove port	http://www.siterequest.com/unavailable
https://%h/unavailable	Specify **https** as protocol, remove port, change path	https://www.siterequest.com/unavailable
www.siterequest.com:8080/%u	Change host name and port, preserve path	http://www.siterequest.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.siterequest.com:554/streamingmedia/sample

Table 4.11 *Sample redirection results*

The example in Figure 4.10 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 {
fallback "https://1.2.3.4:443/%u/unavailable.html"
member 10.12.10.1:80
member 10.12.10.2:80
member 10.12.10.3:80
}
```

Figure 4.10 *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 Chapter 7, *SSL Accelerator Proxies*.
- 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 system 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:
   - If you intend to use the ISAPI filter with all Web sites, select the **ServerName** icon.
   - 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. Select **Properties** from the displayed menu.
6. 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.*
7. Click **Add**.
8. Type a name for the ISAPI filter.
9. Click **Browse** and select the ISAPI filter that you copied in step 1.
10. Click **OK**.

11. 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).
12. 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).
13. Repeat the previous step for any other services that were stopped in step 11.
14. 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.*

Inserting and erasing HTTP headers

When configuring a pool, you can set options to either insert headers into HTTP requests or erase the content of headers from HTTP requests. You configure these options using the pool attributes **header insert** and **header erase**. The following section describe these attributes.

Inserting headers into HTTP requests

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. Optionally, you can include rule variables in the quoted string. For example, the pool definition shown in Figure 4.11 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 4.11 *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**

◆ Note

For more information on these rule variables, see Chapter 5, iRules.

Figure 4.12 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 4.12 *Pool with header insertion string using multiple rule variables*

The above header insertion string inserts the following header 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 4.13 *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 Chapter 5, *iRules* and Chapter 7, *SSL Accelerator Proxies*.

To insert a header into an HTTP request 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, type a string into the **Header Insert** box.
4. Click **Done**.

To insert a header from the command line

To insert a header from the command line, use the following syntax, where **<quoted string>** is the header to insert:

```
b pool <pool_name> header insert <quoted string>
```

Erasing header content from HTTP requests

Another optional attribute of a pool is the **header erase** attribute. Using this attribute, you can configure a pool to erase the contents of a header from an HTTP client request. When you use this attribute, the BIG-IP system erases the contents of the specified header and replaces that content with blank spaces. The header itself is retained.

With this feature, you can erase headers from HTTP requests before forwarding the requests over the network. Such headers might contain sensitive information, such as user IDs or telephone numbers, that must be erased before the information is forwarded.

The client header with the contents to be erased must be specified in the pool definition as a quoted string.

To erase the contents of a client header 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, type a header name into the **Header to Erase** box.
4. Click **Done**.

To erase the contents of a client header from the command line

To erase the contents of a header from the command line, use the following syntax, where **<quoted string>** is the name of the header:

```
b pool <pool_name> header erase <quoted string>
```

Configuring the 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 system 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 system can also apply a rule that sends the traffic to different pools of servers based on the Quality of Service level.

The BIG-IP system 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 system can examine incoming traffic to see if it has a particular QoS or ToS tag in the header. The BIG-IP system 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 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 4.14 *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 Chapter 5, *iRules*.

Configuring the 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 system 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 system can also apply a rule and send the traffic to different pools of servers based on the ToS level.

The BIG-IP system 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 system can examine incoming traffic to see if it has a particular ToS tag in the header. The BIG-IP system can then make a rule-based load balancing decision based on that tag.

Figure 4.15 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 4.15 *Example of a pool that sets the ToS level on a packet*

◆ Note

*If you change the **ip_tos to client** or **ip_tos to server** level on a pool, existing connections continue to use the previous setting.*

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 Chapter 5, *iRules*.

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 Chapter 10, *Address Translation: SNATs, NATs, and IP Forwarding*.

The example in Figure 4.16 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 4.16 *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 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.

To disable SNAT connections that use a specific service

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 Enabling a forwarding pool*, on page 4-58.

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 4.17 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 4.17 *Sample entries in the* ***/config/bigip.conf*** *file*

Figure 4.18 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 4.18 *A rule that disables SNAT connections for a range of ports*

Enabling a forwarding pool

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 4.19 shows an example of a pool configured for forwarding.

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

Figure 4.19 *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, see Chapter 6, Virtual Servers.

Configuring a clone pool

When the **clone pool** attribute is enabled, all network traffic that a pool receives is replicated and sent to a node in the clone pool. The clone pool feature is used for detecting intrusion.

The BIG-IP system can replicate the packets either before or after translating the destination IP address.

You enable the clone pool feature using either the Configuration utility or the **bigpipe pool** command.

Configuring a clone pool using the Configuration utility

1. In the navigation pane, click **Pools**.
2. Click a pool name.
   The properties of the pool are displayed.
3. In the **clone pool** box, select **Before** or **After** from the list box.
4. Click Apply.

Configuring a clone pool from the command line

Use the following command syntax to configure a clone pool:

```
b pool [ clone before | clone after ] <pool_name>
```

5

iRules

- Introducing iRules
- Creating rules
- Understanding rules syntax
- Using rules to select pools
- Using rules to redirect HTTP requests
- Configuring class lists
- Additional rule examples

Introducing iRules

iRules is a powerful and flexible feature that you can use to balance your network traffic. The iRules feature not only allows you to create rules and classes to select pools, but also to configure persistence scenarios that allow the BIG-IP system to search on any type of connection data that you define. Thus, the iRules feature significantly enhances your ability to customize your content-switching to suit your exact needs.

What is a rule?

An element of the iRules feature, a ***rule*** is a user-written script that chooses among two or more load balancing pools. In other words, a rule selects a pool associated with a virtual server. Rules are an optional feature that you can use if you do not want traffic to target the default pool defined for a virtual server. Rules allow you to more directly specify the pools to which you want traffic to be directed.

Once you have created a rule to select a pool, you can then configure that pool to further direct traffic to individual pool members, either to implement persistence or to meet your specific load balancing requirements. When configuring that pool, you can conveniently use the same expression syntax that you use for rules, to specify the expressions that your pool definition might require for persistence. For information on configuring pools, see Chapter 4, *Pools*.

When a packet arrives destined for a virtual server that does not match a current connection, the BIG-IP system can select a pool by evaluating a rule associated with a virtual server. The rule can direct traffic to a pool based on specific data such as IP packet headers, HTTP headers or TCP content. Thus, rules can be configured to ask true or false questions such as:

- Does the packet data contain an HTTP request with a URI ending in **cgi**?
- Does the source address of the packet begin with the octet **206**?
- Does the TCP packet data contain the string **ABC**?

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

Rules are made up of statements and expressions. Within those expressions, you can use many elements, such as functions, operands, literals, and operators. For descriptions of these elements, see *Expressions*, on page 5-4.

A rule example

The rules you create can be simple or complex, depending on your content-switching needs. Figure 5.1 shows an example of a simple rule.

```
if ( http_uri ends_with "gif" or
     http_uri ends_with "html" ) {
   use ( cache_pool )
}
else {
   use ( server_pool )
}
```

Figure 5.1 *Example of a rule*

This simple rule sends .**gif** and **.html** content to pool **cache_pool** and all other content to pool **server_pool**.

Creating rules

You can create rules using either the Configuration utility or the **bigpipe rule** command. Each of these methods is described in this section.

To create a rule using the Configuration utility

1. In the navigation pane, click **Rules**.
   The Rules screen opens.
2. Click the **Add** button.
   The Add Rule screen opens.
3. In the **Name** box, type a 1- to 31-character name.
4. In the **Type** box, select either **Rule Builder** or **Text Input**.
   If you select **Rule Builder**, the BIG-IP system automatically creates a rule for you, based on rule elements that you select or type in the GUI. If you select **Text Input**, a screen appears in which you can type the complete text of your rule.
5. Click **Done**.

To create a rule from the command line

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

```
rule <rule_name> { <if_statement> | <discard_statement> |<use_statement>|<cache_statement>
    | <redirect_statement> |  <log_statement> | <accumulate_statement> }
```

For detailed syntax information on building rule statements, see the remainder of this chapter.

◆ 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 ***Referencing BIG-IP system resources****, on page 6-18.*

Understanding rules syntax

Rules consist of statements, which are made up of a number of different elements, such as functions, constants, variables, and operators. Together, these elements can be used to construct a rule that selects a particular pool associated with a virtual server. They can also be used to redirect client requests.

Rule statements

A rule consists of one or more 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 pool** 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 to** statement sends traffic to a specific destination, rather than to a pool for load balancing.
- A **log** statement logs a message to the Syslog facility. The statement does this by performing variable expansion on the message as defined for the **header insert** pool attribute. For more information on the **header insert** pool attribute, see Chapter 4, *Pools*.
- An **accumulate** statement terminates rules processing until another another packet containing additional data is received from the originating client. This statement is useful with the **http_content** and **tcp_content** rule variables, when not enough data has been received to be successfully evaluated. For information on these rule variables, see *Variables*, on page 5-13.

If not used appropriately, a ***log*** *statement can produce large amounts of output.*

Table 5.1 shows the syntax for rule statements.

Element	Syntax
if statements	`if ( <expression> ) { <statement> }` `[ { else <statement> } ] [ { else if <statement> } ]`
discard statements	`discard`
use pool statements	`use pool ( <pool_name> )`
cache statements	`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> ] [ persist <expression> ]}`
redirect to statements	`redirect to ( <redirect URL> )`
log statements	`log [<facility>.<level>] "<message>"`
accumulate statements	`accumulate`

Table 5.1 *Rule statements and their syntax*

◆ Note

*The maximum number of **if** statements that you can nest in a rule is 100.*

Expressions

Expressions can be specified within rule statements or within pool definitions. Their purpose is to describe some action that the rule or pool is to take, and to provide any data the the rule or pool requires to take the action.

Table 5.2 shows the elements that an expression can contain, and their syntax.

Elements	Syntax

Functions	`findstr()`
	`substr()`
	`getfield()`
	`findclass()`
	`decode_uri()`
	`domain()`
	`imid()`
	`mapclass2node()`
	`node()`
	`wlnode()`

Constant operands	`<regex_literal>` - A regular expression literal that must be a string of 1 to 63 characters enclosed in quotes that can contain regular expressions. `<string_literal>` - A string literal that must be a string of 1 to 63 characters enclosed in quotes. `<address_literal>` - An address literal of the form `<dot_notation_longword> [netmask <dot_notation_longword>]`, where the dot notation longword must be in the form `<0-255>.<0-255>.<0-255>.<0-255>.`

Variable operands	`client_addr`
	`server_addr`
	`client_port`
	`server_port`
	`ip_protocol`
	`link_qos`
	`ip_tos`
	`http_method`
	`http_uri`
	`http_version`
	`http_host`
	`http_header <header tag>`
	`http_cookie <cookie_name>`
	`http_content`
	`http_content_collected`
	`tcp_content`
	`tcp_bytes_collected`

Relational operators	`contains`
	`matches`
	`equals`
	`starts_with`
	`ends_with`
	`matches_regex`
	`one of`
	`redirect to`

Logical operators	`not`
	`and`
	`or`

Cache statement attributes	`origin_pool <pool name>`
	`cache_pool <pool name>`
	`persist <expr>`

Table 5.2 *Elements of an expression and their syntax*

Figure 5.2 is a rule that shows two examples of an expression within a rule. The expressions are indicated in boldface type.

```
rule my_rule {
     if (tcp_content contains "XYZ") {
         use pool xyz_servers
     }
     else if (substr(tcp_content(100), 50, 3)) == "ABC" {
         use pool abc_servers
     }
     else {
         use pool web_servers
     }
}
```

Figure 5.2 *Examples of expressions specified within a rule*

In Figure 5.2, the element **tcp_content** is the variable operand. The element **contains** is the relational operator, and the element **substr()** is the function. The expressions also contain some literals, such as **50**, **3**, **"XYZ"**, and **"ABC"**.

Expressions for use within rules and pools include several important features:

- Expression syntax can be used outside of rules, such as within pool persistence definitions.
- Expressions support several functions for enabling persistence and direct selection of pool members.
- Expression functions, variables, and literals can exist on either side of an operator.
- Expressions allow string concatenation using the plus (**+**) operator.
- Expressions promote integers, addresses, and so on to strings when used with string concatention.
- Functions with no arguments can be specified without parentheses **().**
- The syntax for the **http_header** and **http_cookie** variables matches the syntax for functions (for example, **http_header(<header_tag_string>)**).

The following sections describe all of the elements that an expression can contain.

◆ Note

*The maximum number of **if** statements that you can nest in a rule is 100.*

Functions

There are functions available that you can use as part of expressions. A primary use of such expressions is to specify them within pool definitions, to either return a string for persistence, or to return a node to which the pool can send traffic directly.

Functions that return a string

The following functions return a string that you specify. They are:

- **findstr()** - Finds a string within another string and returns the string starting at the offset specified from the match.
- **substr()** - Returns the string starting at the offset specified.
- **getfield()** - Splits a string on a character and returns the string corresponding to the specific field.
- **findclass()** - Finds the member of a class that contains the result of the specified expression and returns that class member.
- **decode_uri()** - Evaluates the expression and returns a string with any **%XX** escape sequences decoded as per HTTP escape sequences defined in RFC2396.
- **domain()** - Parses and returns up to the specified number of trailing parts of a domain name from the specified expression.
- **imid()** - Used to parse the **http_uri** variable and **user-agent** header field for an i-mode identifier string that can be used for i-mode persistence.

findstr()

The **findstr()** function finds a string within another string and returns the string starting at the offset specified from the match. The function returns an empty string if:

- The argument **<expr>** does not evaluate into a string.
- The string specified by the argument **<string>** is not found in the evaluated expression.
- The specified offset would put the return string outside the evaluated expression.

Syntax

The **findstr()** function takes the following arguments:

```
findstr(<expr>, <string>, <offset>)
findstr(<expr>, <string>, <offset>, <length>)
findstr(<expr>, <string>, <offset>, <termchr>)
```

where:

<expr> is the expression to be searched.

<string> is a literal string to search for. A literal string uses double quotation marks, for example, “user=”.

<offset> is the offset from the found string of the string to return.

<length> is the number of characters of the string to return.

<termchr> is a literal character the string to return is ended at. A literal character uses apostrophes (single quotation marks), for example, ';'.

Examples

This expression returns the string following a query character:

```
findstr(http_uri, "?", 1)
```

This expression returns the three characters following **x=**:

```
findstr(http_uri, "x=", 2, 3)
```

This expression returns the string following **user=** up to the **&** character:

```
findstr(http_uri, "user=", 5, '&')
```

substr()

The **substr()** function returns the string starting at the offset specified. The function returns an empty string if:

- The value of the **<expr>** argument does not evaluate into a string
- The value of the **<offset>** argument puts the return string outside of the evaluated expression.

Syntax

The **substr()** function takes the following arguments:

```
substr(<expr>, <offset>)
substr(<expr>, <offset>, <length>)
substr(<expr>, <offset>, <termchr>)
```

where:

<expr> is the expression providing the source string.

<offset> is the offset from the beginning of the string to return.

<length> is the number of characters of the string to return.

<termchr> is a literal character with which the string to return is ended.

Examples

This expression returns the string starting after the first character:

```
substr(http_uri, 1)
```

This expression returns a string of three characters starting after the second character:

```
substr(http_uri, 2, 3)
```

This expression returns the string starting at the fifth character and ending at the **&** character:

```
substr(http_uri, 5, '&')
```

getfield()

The **getfield()** function splits a string on a character, and returns the string corresponding to the specific field. The function returns an empty string if the value of **<expr>** does not evaluate into a string, or if the specified **<fieldnum>** value is greater than the number of fields the evaluated **<expr>** was split into. If the specified **<split>** character or string is not found in the evaluated **<expr>**, then it is treated as one field.

Syntax

The **getfield()** function takes the following arguments:

```
getfield(<expr>, <split>, <fieldnum>)
```

where:

<expr> is the expression to be split.

<split> is a literal character or string used to split the string into fields. A literal character uses apostrophes (single quotation marks), for example, ';'.

<fieldnum> is the number of the field to return.

Example

The following example of the **getfield** function returns the third field after splitting on a **;** character:

```
getfield(http_uri, ';', 3)
```

Using this example, the expression **ABC;DEF;123** returns the string **123**.

findclass()

The **findclass()** function finds the member of a class that starts with or matches the beginning of that class member and returns the value of that class member. This is similar to the **one of** operator, except that the member is not required to be equal; instead, the member is only required to start with the string and returns the entire member value.

Syntax

The **findclass()** function takes the following arguments:

```
findclass(<expr>, <classname>)
```

where:

<expr> is the expression whose result is searched for in members.

<classname> is a literal class name in which to search for a member.

Example

This expression returns the member containing the domain portion of the **http_host** string:

```
findclass(domain(http_host, 2), external_servers)
```

decode_uri()

The **decode_uri()** function evaluates the expression and returns a string with any **%XX** escape sequences decoded as per HTTP escape sequences defined in RFC2396.

Syntax

The **decode_uri()** function takes the following arguments:

```
decode_uri(<expr>)
```

where:

<expr> is the expression providing the string to convert.

Example

This expression returns the HTTP URI with any escape sequences decoded:

```
decode_uri(http_uri)
```

This expression returns the binary string **0x00 0x03**:

```
decode_uri("%00%03")
```

domain()

The **domain()** function parses and returns up to the specified number of trailing parts of a domain name from the specified expression. The function automatically parses out the portion following a ':' (representing a port or service) from the return string. The function also detects a dotted quad and returns the full quad regardless of the **<count>** specified. If the evaluated domain name does not have the specified number of trailing parts, the number of parts found are returned.

Syntax

The **domain()** function takes the following arguments:

```
domain(<expr>, <count>)
```

where:

<expr> is the expression to be parsed for a domain name.

<count> is the number of trailing parts in the domain name to return.

Example

The following example of the **domain** function returns the last two parts of a domain name:

```
domain(http_host, 2)
```

Using this example, the host name **www.mysite.com** returns the string **mysite.com**.

imid()

The **imid()** function is specifically used to parse the **http_uri** variable and **user-agent** header field for an i-mode identifier string that can be used for i-mode persistence. The **imid()** function takes no arguments and simply returns the string representing the i-mode identifier or the empty string, if none is found.

Functions that return a node address

There are a number of expressions that you can use primarily to directly select a particular node (pool member) within a pool. They are:

- **node()** - Returns a literal node address converted from a string representation of an address and port.
- **mapclass2node** - Represents a short-hand combination of the functions **findclass()**, **findstr()**, and **node()**.
- **wlnode()** - Returns a literal node address converted from a specific BEA WebLogic™ string format, representing the application IP address and service, into a literal node address.

node()

The **node()** function returns a literal node address converted from a string representation of an address and port. This function is designed primarily to be used with the persist expressions for directly selecting a node to which to persist.

Syntax

The **node()** function takes the following arguments:

```
node(<expr>)
```

where:

<expr> is the expression providing the string to convert.

Examples

```
node("10.0.0.3:80")     // returns the node represented by
10.0.0.3, port 80

node(getfield(http_cookie("SERVER"), ';', 1))
```

mapclass2node()

The **mapclass2node()** function is a short-hand combination of the functions **findclass()**, **findstr()**, and **node()**. The function is equivalent to the following syntax:

```
node(findstr(findclass(<expr>, <classname>), "",1))
```

where "" represents a space character.

Syntax

The **mapclass2node()** function takes the following arguments:

```
mapclass2node(<expr>, <classname>, [<delim>])
```

where:

<expr> is the expression whose result is searched for in members of the named class.

<classname> is a literal name for the class in which to search for a member.

<delim> is the delimiter separating the matched string from the node address and service.

Examples

For the following class:

```
class external_servers {
     "ABC 10.0.0.1:80",
     "DEF 10.0.0.2:80"
}
```

this example returns a node that matches one of the server names found in the class **external_servers**:

```
mapclass2node(findstr(http_uri, 'server=', 7, 6), external_servers)
```

wlnode()

The **wlnode()** function returns a literal node address converted from a string of the BEA WebLogic format for identifying the application node to which the connection should persist.

The **wlnode()** function is designed primarily to be used with the persist expressions for directly selecting a node to which to persist.

Syntax

The **wlnode()** function takes the following arguments:

```
wlnode(<expr>)
```

where:

<expr> is the expression providing the BEA WebLogic string.

Examples

```
wlnode("10.0.0.3:80")     // returns the node represented by
                          10.0.0.3, port 80

wlnode(http_cookie("JSESSIONID"))
```

Constant and variable operands

Operands are of two types--constant operands (constants) and variable operands (variables). You can use both types of operands in an expression.

Constants

Constants are elements of an expression whose values do not change. Possible constants 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

Variables

Variables are elements of an expression that have changeable values. Therefore, 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. Following are descriptions of the variables that you can use within a rule.

IP packet header variable

Table 5.3 lists and describes the available IP packet header variables for use within a rule.

Variable Name	Description
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.

Table 5.3 *IP packet header variables*

HTTP request string variables

The BIG-IP system replaces all HTTP request string variables with string literals. In command line syntax, you refer to HTTP request variables 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 or content data. Before an HTTP request variable is used in a relational expression, the BIG-IP system replaces the variable with the extracted string.

Table 5.4 lists and describes the available HTTP request string variables for use within a rule.

Variable Name	Description
http_method	The **http_method** variable specifies the action of the HTTP request. Common values are **GET** or **POST**.
http_uri	The **http_uri** variable specifies the URL, but does not include the protocol and the fully qualified domain name (FQDN). For example, if the URL is **http://www.mysite.com/buy.asp**, then the URI is **/buy.asp**.
http_version	The **http_version** variable specifies the HTTP protocol version string. Possible values are "**HTTP/1.0**" or "**HTTP/1.1**".
http_host	The **http_host** variable specifies 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 variable specifies 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 **http_header** variable 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 want 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 ) }`

Table 5.4 *HTTP requeset string variables*

Variable Name	Description

http_content[(<minlength>)]	The **http_content** variable evaluates the string following an HTTP content tag that you specify. For example, if you want to load balance traffic based on the value of the string **date,** the rule statement might look as follows:
	`if (findstr (http_content, "date=",5,6) == "052102" {`
	`  use pool date_pool;`
	`} else if (http_content_collected < 512)`
	`  accumulate;`
	`} else {`
	`  use pool other_pool`
	`}`
	Note the following:
	The **http_content** variable uses the **content_length** or **transfer_encoding** fields to determine the available content. If these headers cannot be found, the **http_content** variable simply returns the amount received and does not know if there is more data to be sent. If not enough data is found to meet the specified minimum required, the connection can hang. To avoid this, we recommend using the **accumulate** statement and the **http_content_collected** variable in any rule containing the **http_content** variable.
	The system uses a buffer space of 16384 bytes to store all HTTP-collected content, including headers. Therefore, headers reduce the available buffer space for the data that the **http_content** variable returns.

http_content_collected	The **http_content_collected** variable returns the amount of content that has currently been collected.

Table 5.4 *HTTP requeset string variables*

TCP string variables

The BIG-IP system replaces all TCP request string variables with string literals. In command line syntax, you refer to TCP request variables by a predefined set of names. Internally, a TCP request variable points to a method for extracting the desired string from the current TCP request data. Before a TCP request variable is used in a relational expression, the BIG-IP system replaces the variable with the extracted string.

Table 5.5 lists and describes the available TCP request string variables for use within a rule.

Variable Name	Description
tcp_content	The **tcp_content** variable allows you to create a basic expression that load balances traffic based on arbitrary data within a TCP/IP connection. The variable returns TCP data content up to the BIG-IP system's maximum application buffer size (currently 16384 bytes). The most likely use of the **tcp_content** variable is to search for specific strings when establishing a non-HTTP connection. If the **tcp_content** variable does not receive the amount of minimum data required, the connection hangs until the timeout expires. To avoid this, we recommend that you carefully use the **accumulate** statement and the **tcp_butes_collected** variable in any rule containing the **tcp_content** variable.
tcp_bytes_collected	The **tcp_bytes_collected** variable returns the amount of content that has currently been collected.

Table 5.5 *TCP request string variables*

Operators

Operators can be of two types--relational operators and logical operators.

Relational operators

In a rule, ***relational operators*** compare two operands to form relational expressions. Table 5.6 lists the types of expressions you might want to use, and their corresponding relational operators.

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 5.6 *The relational operators*

Logical operators

In a rule, ***logical operators*** modify an expression or connect two expressions together to form a logical expression. Table 5.7 lists the types of expressions you might want to use and their corresponding logical operators.

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 5.7 *The logical operators*

Using rules to select pools

Rules can search for various kinds of data within a request before selecting the appropriate pool to service that request. Table 5.8 lists the various criteria you can use when creating a rule to select a pool.

Pool-selection criteria	Description
Header or content data	You can send connections to a pool or pools based on header or content information that you specify.
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.
***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.
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 5.8 *Criteria for selecting a pool through a rule*

◆ **Note**

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

Selecting pools based on header or content data

Using rule variables such as **http_header**, **http_content**, and **tcp_content**, you can create a rule that selects a pool based on data that resides in the header or content of a request.

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 5.3 shows a rule that illustrates this example.

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

Figure 5.3 *Sample rule using an HTTP request string variable*

Regarding the **tcp_content** variable, you might want a rule that logically states: "If the packet data contains a TCP request containing the string "XYZ", then load balance using the pool **xyz_servers**. If the string is not found, search for the string "ABC" at the specified offset and load balance using the pool **abc_servers**. If the string "ABC" is not found, load balance using the pool **web_servers**".

Figure 5.4 shows a rule that illustrates this example.

```
rule my_rule {
     if (tcp_content contains "XYZ") {
         use pool xyz_servers
     }
     else if (substr(tcp_content(100), 50, 3)) == "ABC" {
         use pool abc_servers
     }
     else {
         use pool web_servers
     }
}
```

Figure 5.4 *Sample rule using a TCP request string variable*

Another example is shown in Figure 5.5.

```
rule my_rule {
   if (http_content contains "ABC") {
      use pool http_pool
   }
   else if (http_content_collected < 20) {
      accumulate
   }
   else {
      discard
   }
}
```

Figure 5.5 *Sample rule using an HTTP request string variable*

Note that in Figure 5.5, the search for string "ABC" is not limited to the first 20 bytes in the HTTP content. If you want to restrict a search to a specific region of the content, such as the first 20 bytes, you can include an ecpression in the rule, using the **substr()** function. Figure 5.6 shows an example.

```
if (substr(http_content, 0, 20) contains "ABC") {
      use pool http_pool
   }
   else if (http_content_collected < 20) {
      accumulate
   }
   else {
      discard
   }
}
```

Figure 5.6 *Sample rule that restricts a search to the first 20 bytes of data*

For information on functions such as the **substr()** function shown in the above example, see *Functions*, on page 5-7.

To specify HTTP request string or TCP request string 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, type a unique name for the rule.
4. In the **Type** box, click the button for **Rule Builder**.
5. Click **Next**.
6. On the left side of the screen, select a variable.
7. Fill in all information within the row.
8. Click **Next**.
9. Select **Discard**.
10. Click **Next**.
11. Select either **No Action** or **Discard** from the box.
12. Click **Done**.

To specify HTTP request string or TCP request string variables within a rule from the command line

The following example shows the syntax for specifying an IP packet header variable within a rule.

```
b rule my_pool { if ’( http_uri ends_with "cgi" ) { use ( cgi_pool ) }\
    else { use ( another_pool ) } }
```

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 7-54.

Selecting pools based on IP packet header data

In addition to specifying the HTTP and TCP request string variables within a rule, you can also select a pool by specifying IP packet header information. 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

The following sections describe these specific types of IP packet header data that you can specify within a rule. For the procedure on specifying IP packet header variables within a rule, see *To specify IP packet header variables within a rule using the Configuration utility*, on page 5-23.

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 5.7 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 pool clients_from_206
     }
      else {
         use pool other_clients_pool
       }
}
```

Figure 5.7 *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 5-34.

Port numbers

The BIG-IP system 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 5.8.

```
rule my_rule {
    if (client_port > 1000) {
        use pool slow_pool
     }
      else {
         use pool fast_pool
       }
}
```

Figure 5.8 *A rule based on a TCP or UDP port number*

IP protocol numbers

The BIG-IP system 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 5.9.

```
rule my_rule {
    if (ip_protocol == 6) {
        use pool tcp_pool
     }
      else {
         use pool slow_pool
       }
}
```

Figure 5.9 *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 system 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 5.10.

```
rule my_rule {
  if (link_qos > 2) {
    use pool fast_pool
  }
  else {
    use pool slow_pool
  }
}
```

Figure 5.10 *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 *Configuring the Quality of Service (QoS) level*, on page 4-56.

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 system 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 5.11.

```
rule my_rule {
  if (ip_tos == 16) {
    use pool telnet_pool
  }
  else {
    use pool slow_pool
  }
}
```

Figure 5.11 *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 *Configuring the Type of Service (ToS) level*, on page 4-57.

Specifying IP packet header variables within a rule

You can use either the Configuration utility or the **bigpipe rule** command to specify IP packet header variables within a rule.

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. On the left side of the screen, select a variable.
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**.

To specify IP packet header variables within a rule from the command line

The following example shows the syntax for specifying an IP packet header variable within a rule.

```
b rule my_pool { if ’( client_addr == 10.12.12.10 )’ { use ’( pool_A80 )’ } }
```

Selecting pools based on the one of operator

The BIG-IP system 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.

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 5.12, 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 pool aol_pool
     }
 else {
    use pool all_pool
 }
}
```

Figure 5.12 *Example of a rule without the* ***one of*** *operator*

Using the **one of** operator instead, you can cause the BIG-IP system 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 5.13 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 pool aol_pool
     }
      else {
         use pool all_pool
       }
}
```

Figure 5.13 *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.

The following sections describe the types of classes that the BIG-IP system allows and the commands for creating classes.

Selecting pools 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 system does not do any additional load balancing or persistence processing.

Figure 5.14 shows an example of a rule containing a cache statement.

```
rule my_rule {
    if ( http_host starts_with "abc" ) {
        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 pool host named_servers
    }
}
```

Figure 5.14 *An example of a rule containing a cache statement*

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. Figure 5.15 shows an example of a cache statement within a rule.

```
cache(http_method == "GET") {
   origin_pool pool1
   cache_pool pool2
   persist (domain(http_host, 2) + http_uri)
}
```

Figure 5.15 *Example of a cache statement within a rule*

Table 5.9 describes the **cache** statement attributes and their syntax.

Attribute	Description	Required?
origin_pool <pool_name>	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 system to redirect a missed request from a cache.	Yes
cache_pool <pool_name>	Specifies a pool of cache servers to which requests are directed to optimize cache performance.	Yes
hot_pool <pool_name>	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.	No

Table 5.9 *Description of **cache** statement syntax*

Attribute	Description	Required?
persist <expr>	Specifies an expression that will be evaluated and used to persist to the same node within the cache pool.	No
hot_threshold <hit_rate>	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**).	No
cool_threshold <hit_rate>	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.	No
hit_period <seconds>	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.	No
content_hash_size <sets_in_content_hash>	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.	No

Table 5.9 *Description of* ***cache*** *statement syntax*

Using rules to redirect HTTP requests

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 system. 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 5.16 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 pool web_pool
       }
}
```

Figure 5.16 *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), 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 *Redirecting HTTP requests*, on page 4-48.

Configuring class lists

Class lists are useful for creating rules. Specifying the **one of** operator and a class name within a rule eliminates the need to specify multiple values for a variable within the expression. For a more detailed explanation, see *Selecting pools based on the one of operator*, on page 5-23.

The remainder of this section describes the types of class lists that you are allowed to create, the storage options available, some predefined classes included in the BIG-IP system, and the procedure for creating a class.

Class types

When using the **one of** operator within a rule, you can specify any of three types of classes--classes of IP addresses, classes of strings, and classes of numeric values.

IP address classes

There are two types of IP address classes--network IP address and host IP address.

The following command creates a network IP address class named **my_netwk** and contains network IP addresses:

```
b class my_ntwk { network 10.2.2.0 mask 255.255.255.0 }
```

Figure 5.17 shows the resulting network IP address type of class.

```
class my_netwk {
  network 10.2.2.0 mask 255.255.255.0
}
```

Figure 5.17 *An example of a network IP address type of class*

The following command creates a host IP address class named **my_host** and contains one or more host IP addresses:

```
b class my_host { host 10.1.1.1 }
```

Figure 5.18 shows the resulting host IP address type of class.

```
class my_host {
  host 10.1.1.1
}
```

Figure 5.18 *An example of a host IP address type of class*

String classes

The following command creates a string class named **images**.

```
b class images { \".gif\" }
```

Note: *This example shows the use of escape characters for the quotation marks.*

Figure 5.19 shows the resulting string type of class.

```
class images {
  ".gif"
}
```

Figure 5.19 *An example of a string type of class*

Numeric value classes

The following command creates a numeric value class named **my_protos**.

```
b class my_protos { 27 38 93 }
```

Figure 5.20 shows the resulting numeric value type of class.

```
class my_protos {
    27
    38
    93
}
```

Figure 5.20 *An example of a numeric value type of class*

Note

The size of a class is limited by system resources only.

Storage options

The BIG-IP system allows you to store classes in two ways, either in-line or externally.

In-line storage

Classes that are stored in-line are saved in their entirety in the **bigip.conf** file. Also, when any data in the class needs to be updated, the entire class must be reloaded. In general, in-line storage uses additional system resources due to extensive searching requirements on large classes. Also, in-line storage requires you to reload entire classes when incrementally updating data.

To create an in-line class using the Configuration utility

1. In the navigation pane, click **Rules**.
2. Click the Classes tab.
   This displays a list of the currently-defined in-line classes.
3. Click the **Add** button.
   This displays the Add Class screen.
4. In the **Class Name** box, type in a class name.
5. In the **Class Type** box, select a class type, either **Address**, **String**, or **Value**.
6. In the **File Mode** box, select a file mode, either **Read** or **Read/Write**.
7. In the **Sizing** box, type a number.
8. Click **Done**.

External storage

You have the option to store classes in another location on the BIG-IP system, that is, outside of the **bigip.conf** file. Such classes are called ***external classes***. The default location for storing external classes is the **/config** directory. Because the class is stored externally in another location, the **bigip.conf** file itself contains only meta-data for the class. The data in an externally-stored class file is stored as a comma-separated list of values (CSV format).

Creating external classes is useful because data does not need to be sorted when being loaded. Instead, data is stored in a hash-table in the kernel. This storage method translates to improvements in performance when a rule uses a large class to direct traffic to a pool.

The syntax for the class meta-data that is stored in the **bigip.conf** file is as follows:

```
class <name> extern {
<variable> <value>
<variable> <value>
...
}
```

The value of the **<name>** field is the name that you want to assign to the class.

Table 5.10 lists and describes the allowed values for the **<variable>** field, along with the corrpesonding values for the **<value>** field. Note that the value of the **<value>** field varies depending on the whether the **<variable>** field is **filename**, **type**, **mode**, or **sizing**.

<variable> Values	Description	<value> Values
filename	The location of the externally-stored class	Any pathname
type	One of the three allowed class types	**string**, **value**, or **ip**
mode	A permission value that determines whether or not the BIG-IP system can write to the externally-stored class file during a **save** operation. If the mode is set to **read**, the class file cannot be modified during a **save** operation or deleted when a class is removed.	**read** and **readwrite**
sizing	The expected size of the class. This value is used when creating the kernel hash table for the class. The actual class size can be larger or smaller than the **sizing** value.	Any number

Table 5.10 *Allowed values within* ***bigip.conf*** *meta-data for an externally-stored class*

Figure 5.21 shows an example of class meta-data in the **bigip.conf** file. Note that this meta-data references the externally-stored class file **/home/ip2.class**.

```
class ipvals extern {
    filename /home/ip2.class
    type ip
    mode readwrite
    sizing 10000
}
```

Figure 5.21 *An example of class meta-data in the* ***bigip.conf*** *file*

The data in an external class file is stored in comma-separated lists, and the formats of any data values, such as IP addresses, match the formats used in the **bigip.conf** file. Continuing with the example above, Figure 5.22 shows the contents of the class file **/home/ip2.class**.

```
network 195.93.32.0 mask 255.255.255.0
,network 195.93.33.0 mask 255.255.255.0
,network 195.93.34.0 mask 255.255.255.0
,network 195.93.48.0 mask 255.255.255.0
,network 195.93.49.0 mask 255.255.255.0
,network  195.93.50.0 mask 255.255.255.0
```

Figure 5.22 *An example of an external class file*

Creating and deleting external classes

Using the Configuration utility, you can create or delete externally-stored classes.

To create an external class using the Configuration utility

1. In the navigation pane, click **Rules**.
2. Click the External Classes tab.
   This displays a list of the currently-defined external classes.
3. Click the **Add** button.
   This displays the Add External Class screen.
4. In the **Class Name** box, type in a class name.
5. In the **Class Type** box, select a class type, either **Address**, **String**, or **Value**.
6. In the **File Name** box, type the path name to the class.
7. In the **File Mode** box, select a file mode, either **Read** or **Read/Write**. The default value is **Read**.
8. In the **Sizing** box, type a number. The default value is **1024**.
9. Click **Done**.

To delete an external class using the Configuration utility

1. In the navigation pane, click **Rules**.
2. Click the External Classes tab.
   This displays a list of the currently-defined external classes.
3. In the **Delete** column, click the trashcan corresponding to a class name.

◆ Note

*If you use the **bigpipe reset** command to delete configuration components, the command does not delete any externally-stored classes. You must delete externally-stored classes explicitly, using the Configuration utility as shown above.*

Displaying external class data

Using either the Configuration utility or the bigpipe command, you can display the properties of an existing external class, or you can display either the class meta-data or the contents of the external class.

To display the properties of an external class using the Configuration utility

1. In the navigation pane, click **Rules**.
2. Click the External Classes tab.
   This displays a list of the currently-defined external classes.
3. In the **Class Name** column, click a class name.

To display external class meta-data from the command line

You can display the meta-data information for an externally stored class using the following **bigpipe** command syntax, where **<name>** is the class name.

```
b class <name> show
```

To display the contents of an external class from the command line

You can display the contents of an external class file using the following **bigpipe** command syntax, where **<name>** is the class name.

```
b class <name> dump
```

To display a class member from the command line

You can display the contents of a particular class member using the following **bigpipe** command syntax, where **<name>** is the class name and **<value>** is the name of the member.

```
b class <name> member show <value>
```

Managing external class members

From the properties page of an external class, you can add, delete, or search for class members. This abilty to manage class members eliminates the need to re-load the entire class list when the class needs to change incrementally.

◆ Note

*You can only add or delete a class member when the class mode is set to **ReadWrite**. To change the class mode, use the Configuration utility.*

To manage external class members using the Configuration utility

1. In the navigation pane, click **Rules**.
2. Click the External Classes tab.
   This displays a list of the currently-defined external classes.
3. In the Class Name column, click a class name.
4. Enter the appropriate member data.
5. In the **Resources box**, press the **ADD**, **DEL**, or **FIND** button.
   Pressing one of these buttons causes the change to take effect immediately.

To manage external class members from the command line

You can add or delete a member from an external class using the following **bigpipe** command syntax, where **<name>** is the class name and **<value list>** is a list of one or more elements of the appropriate type (**Address**, **String**, or **Value**).

```
b class <name> member add | delete { <value list> }
```

For example, the following command adds the member **".gif"** to the class **my_class**:

```
b class my_class member add \".gif\"
```

Including external class lists when synchronizing configurations

When you synchronize a BIG-IP system configuration for redundant systems, the BIG-IP system only includes class lists that reside in the **/config** directory (that is, in-line stored class lists). If you want the synchronization to include externally-stored class lists, you must add anassociated entry for each class list into the **bigip.conf** file. You add these entries using the **bigpipe db** command.

To include external class lists when synchronizing configurations

When you want the BIG-IP system to include externally-stored class lists when synchronizing configurations for redundant systems, use the **bigpipe db** command to create special entries in the **bigip.conf** file. For example,

the following command adds an entry into the **bigip.conf** file for the externally-stored class list **/home/string/class**. The **700** number designates the entry as corresponding to an externally stored class list.

```
b db set Common.Bigip.Sync.700.file = "/home/string/class"
```

Predefined classes

In addition to the **one of** operator, the BIG-IP system includes a number of predefined classes for you to use with this operator. They are:

- AOL Network
- Image Extensions
- Non-routable addresses

These classes 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
```

Additional rule examples

This section contains additional examples of rules including:

- Cookie rule
- Language rule
- AOL rule
- Cache rule
- Protocol specific rule

Cookie rule

Figure 5.23 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 pool virtual_pool
}
else {
   use pool other_pool
}
```

Figure 5.23 *Cookie rule example*

Language rule

Figure 5.24 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 pool french_pool
   }
   else if ( http_header "Accept-Language" equals "sp" ) {
         use pool spanish_pool
   }
   else {
   use pool english_pool
}
```

Figure 5.24 *Sample rule that load balances based on the language requested by the browser*

AOL rule

Figure 5.25 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 pool aol_pool
   }
   else {
      use pool other_pool
   }
}
```

Figure 5.25 *First 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 pool aol_pool_https
   }
   else {
      use pool 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 5.26 *Second example of an AOL rule*

Cache rule

Figure 5.27 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 pool cache_pool
}
else {
   use pool server_pool
}
```

Figure 5.27 *An example of a cache rule*

Rule using the ip_protocol variable

Figure 5.28 shows a rule that uses the **ip_protocol** variable.

```
rule myrule {
   if ( ip_protocol == 37 ) {
      use pool bootp_pool
   } else if ( ip_protocol == 22 ){
      use pool ipsec_pool
   }
    else {
       use pool slow_pool
   }
  }
```

Figure 5.28 *An example of an IP protocol rule*

Rule using IP address and port variables

Figure 5.29 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 pool fast_pool
   } else if ( server_port equals 80 ){
      use pool fast_pool
   }
    else {
       use pool slow_pool
   }
  }
```

Figure 5.29 *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 5.30.

```
class images {
  ".gif"
  ".jpg"
  ".bmp"
}
```

Figure 5.30 *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 5.31 shows this rule.

```
rule myrule {
   if ( http_uri ends_with one of images ) {
     use pool image_pool
   }
   else {
      use pool dynamic_pool
   }
```

Figure 5.31 *Example of a rule using the one of operator*

Rule 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 4-53.

◆ Note

The variables that the SSL Acceleratory proxy uses to insert headers into HTTP requests cannot be used within pool and rule expressions.

6

Virtual Servers

- Introducing virtual servers
- Virtual server types
- Virtual server options
- Using other BIG-IP system features

Introducing 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.)

Before creating a virtual server, you must configure a load balancing pool. You can then create the virtual server, specifying that pool as the destination for any traffic coming from that virtual server. Furthermore, if you want some of the traffic from that virtual server to go to a pool other than the one specified, you can create a rule that directs certain types of traffic to that other pool.

In order to configure virtual servers, you need to know:

- Which virtual server type meets your needs
- Whether you need to enable certain virtual server options

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 6.1 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 6.1 *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 system, 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, Post-Setup Tasks. For information on disabling a virtual server for a specific VLAN, see ***Enabling or disabling a virtual server****, on page 6-11.*

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 system. 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 system receives a connection request for that site, the BIG-IP system 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 system 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 system. For example, when the BIG-IP system 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 system 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.

A wildcard virtual server is enabled for all VLANs by default. 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 6-5.

Creating wildcard virtual servers

Creating a wildcard virtual server requires three tasks. 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 procedures describe these tasks, using both the Configuration utility and the command line.

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 Chapter 4, *Pools*.

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 6-3.
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, click **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

You can define multiple wildcard virtual servers that run simultaneously. Each wildcard virtual server must be assigned to an individual VLAN, and therefore can handle 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

1. In the Navigation pane, click **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 6.1 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 6.1 *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 system 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 6.2.

```
bigpipe virtual 192.168.1.0:http {
    netmask 255.255.255.0 broadcast 192.168.1.255
    use pool default_webservers
}
```

Figure 6.2 *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 **Enable Arp** check box.
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 system to load balance connections across transparent devices. You can create another wildcard virtual server on

the other side of the BIG-IP system to forward packets to virtual servers receiving connections from the transparent devices and forwarding them to their destination.

◆ Tip

*If you do not want the BIG-IP system 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 Chapter 4, Pools.*

◆ 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, Post-Setup Tasks. For information on disabling a virtual server for a specific VLAN, see **Enabling or disabling a virtual server**, on page 6-11.*

Virtual server options

For each type of virtual server, you can configure several options. These options are shown in 6.2.

Option	Description
Display virtual server information	Using the **bigpipe virtual show** command, you can display information about one or more virtual servers.
Enable/disable a virtual server	By default, a virtual server is enabled. If you want to disable a virtual server, you can do so.
Enable/disable a virtual address	By default, a virtual address is enabled. If you want to disable a virtual address, you can do so.
*Override netmask and broadcast	*You can override the default netmask and broadcast for a network virtual address.
Enable/disable address and port translation	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.
Reset onnections when service is down	You can configure the BIG-IP system to automatically reset connections to the virtual server when the service is down.

***Table 6.2** Virtual server configuration options*

Option	Description
Enable/disable 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.
Disable ARP requests	You can control gratuitous ARPs and ARP requests.
Enable/disable TCP Resets on timeout	When you enable this option, the virtual server sends a TCP Reset when a TCP connection is timed out. The default setting for this option is **Enabled**.
Set FastFlow packet acceleration	You can speed up packet flow for TCP connections when the packets are not fragmented.
Set a connection limit	You can set a concurrent connection limit on one or more virtual servers.
Mirror state 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.
Define a last hop pool	You can direct reply traffic to the last hop router using a last hop pool. This overrides the **auto_lasthop** setting.
Define resources	You can configure a virtual server to use a pool or reference a rule. You can also specify that the virtual server is to function as a forwarding virtual server.
Load balance traffic for any IP protocol	You can configure a virtual server to load balance traffic that uses IP protocols other than TCP and UDP. For example, you can load balance IPSEC traffic from virtual private networks (VPNs).
Delete a virtual server	You can delete an existing virtual server.
Reset virtual server statistics	You can reset virtual server statistics.

Table 6.2 *Virtual server configuration options*

Displaying information about virtual addresses

You can 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 system was last rebooted, or since the BIG-IP system became the active unit (redundant configurations only).

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 to a **down** state.

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
```

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 system, 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 system 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 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 system 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
```

Resetting connections when a service is down

Using either the Configuration utility or the **bigpipe virtual** command, you can configure the BIG-IP system to automatically reset connections to the virtual server when the service becomes unavailable.

To reset connections using the Configuration utility

1. In the navigation pane, click **Virtual Servers**.
2. In the list of virtual servers, click a virtual server name.
3. Click the Virtual Server Properties tab.
4. Click the **Enable Reset on Service Down** check box.
5. Press the **Apply** button.

To reset connections from the command line

Use the following command-line syntax:

```
b virtual <ip address>:<service> svc_down reset enable
```

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

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
```

Disabling ARP requests

By default, the BIG-IP system responds to ARP requests for the virtual server address and sends a gratuitous ARP request for router table updates.

To disable ARP requests 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. Clear the **Enable ARP** check box.
4. Press the **Apply** button.

To disable ARP requests from the command line

To disable ARP requests from the command line, use the following command-line syntax:

```
b virtual <ip address>:<service> arp disable
```

Disabling software acceleration 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 enabled 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
```

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
```

Mirroring virtual server state

Mirroring provides seamless recovery for current connections when a BIG-IP system 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 system performance and is primarily useful for long-lived services like FTP and Telnet. Mirroring is not as useful for short-lived connections like HTTP.

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

Important

Connection mirroring is not supported on virtual servers that are the targets of an SSL or Akamaizer Proxy.

To control mirroring for a virtual server

Use the **bigpipe virtual mirror** command to enable or disable mirroring of connections. The syntax of the command is:

```
b virtual <virt addr>:<service> mirror conn enable | disable
```

To display 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 system 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
```

Setting up last hop pools for virtual servers

In cases where you have more than one router sending connections to a BIG-IP system, 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
```

Referencing BIG-IP system resources

Once you have created a pool or a rule, you must configure the virtual server to reference the pool or rule. You can configure a virtual server to reference a pool or rule by using either the Configuration utility or the **bigpipe** command.

To configure a virtual server that references a pool or 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 **Pool** or **Rule**.
4. In the list of pools or rules, select the pool or 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 6.3.

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 6-9.
<rule_name_reference>	A rule name reference. Rule names are strings of 1 to 31 characters. `use rule <rule_name>`

Table 6.3 *The command line rule elements*

Load balancing traffic for any IP protocol

With the BIG-IP system, you can configure a virtual server to load balance IP traffic other than TCP and UDP traffic. This option can be configured on both translating and non-translating virtual servers.

One benefit of this feature is that you can load balance virtual private network (VPN) client connections across several VPNs, eliminating the possibility of a single point of failure. A typical use of this feature is for load balancing multiple VPN gateways in an IPSEC VPN sandwich, using non-translating virtual servers.

An important point to note, however, is that although address translation of such protocols can be optionally activated, some protocols, such as IPSEC in AH mode, rely on the IP headers remaining unchanged. In such cases, you should use non-translating network virtual servers.

By default, this feature is disabled on a virtual server.

To enable load balancing for any IP protocol using the Configuration utility

1. In the navigation pane, click **Virtual Servers**.
2. In the **Virtual Server** list, click the virtual server for which you want to enable any IP load balancing.
3. Click the Virtual Address Properties tab.
4. In the **Any IP Traffic** box, click the **Enable** check box.
5. Click **Apply**.

To enable load balancing for any IP protocol from the command line

Use the **any_ip** option with the **bigpipe virtual** command, as shown in the following syntax:

```
b virtual <ip address:service> any_ip enable
```

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 system features

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 Chapter 10, *Address Translation: SNATs, NATs, and IP Forwarding*. For details on persistence for connections that should return to the node to which they last connected, see Chapter 4, *Pools*.

7

SSL Accelerator Proxies

- What is an SSL Accelerator proxy?
- Creating an SSL Accelerator proxy
- Authentication
- Encryption and decryption
- Authorization
- Network traffic control
- Other SSL protocol options

What is an SSL Accelerator proxy?

An SSL Accelerator proxy enables the BIG-IP system to accept and terminate any connections that are sent by way of a fully SSL-encapsulated protocol. As an option, you can configure an SSL proxy to also initiate secure connections to a target web server.

Examples of fully-encapsulated SSL protocols that the BIG-IP system supports are: HTTPS, SMTPS, NNTPS, TELNETS, IMAPS, IRCS, POP3S, and FTPS.

In accepting and terminating SSL connections, the SSL proxy performs many functions. Not only can the proxy off load certificate verification tasks and encryption/decryption tasks from a target web server, it can also intelligently control the destination of client requests, for the purposes of authorization and load balancing.

Some of the features of the SSL proxy are:

- An ***SSL-to-server*** feature, which enables the proxy to establish secure connections between the proxy and a target web server
- A FIPS-140 Level-3 compliant, tamper-proof mechanism for storing private keys
- Support for certificate revocation lists (CRLs)
- Certificate-based client authorization using a Light-weight Directory Access Protocol (LDAP) server.

When you create a proxy, many of the configuration options include default values. These default values allow you to use the proxy as is, or customize it to further suit your particular needs.

Figure 7.1 shows how an SSL proxy fits into a standard client-server environment. The figure shows that a proxy can accept a request from a client, forward the request on to a content server, receive the response, and then forward the response back to the client.

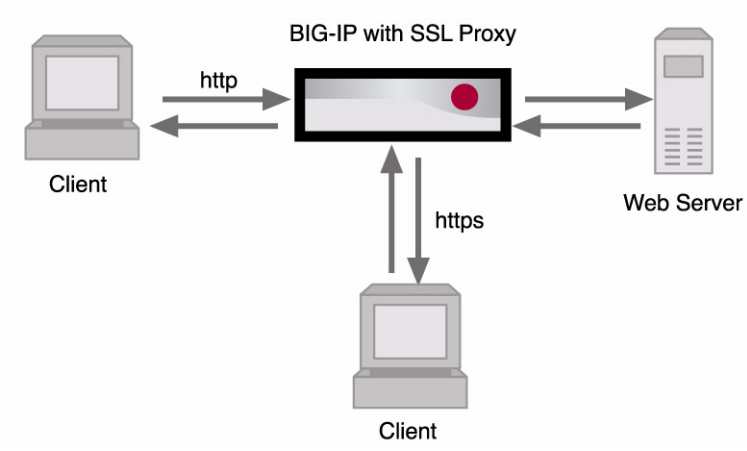

Figure 7.1 *An SSL Accelerator proxy on the BIG-IP system*

Summary of features

An SSL proxy includes several important features. Table 7.1 summarizes these features.

Features	Description
Graphical or command line interface	Through the Configuration utility or the **bigpipe proxy** command, you can create an SSL proxy that includes default values for its configuration options. You can either use the default values or change them to suit the needs of your networking environment.
Authentication	**Validation**--An SSL proxy can perform the same certificate validation tasks that are normally performed by clients and servers when processing SSL connections. By validating client and server certificates, the proxy allows you to off load this task from your target web servers to the BIG-IP system, thereby enhancing the availability of your web servers. **Verification--**In terminating SSL connections, an SSL proxy can perform a full range of certificate verification tasks, including the verification of both client and server certificates. For example, you can also define the extent to which the proxy verifies client certificates--you can configure the proxy to request client certificates, require them, or ignore them altogether. Also, the SSL proxy provides additional security through its ability to encrypt and decrypt proxy-to-server connections and through its provision of tamper-proof mechanisms for private key storage. **Revocation**--When a client or server presents a certificate to the proxy, the proxy can check a list of revoked certificates (known as a CRL) as part of the certificate verification process.
Encryption and decryption	As a way to off load work from target web servers, the SSL proxy decrypts incoming client requests. As an added security option, you can configure the proxy to re-encrypt a request before forwarding it on to a server. Encryption and decryption of requests and responses are based on particular ciphers that you specify as part of proxy configuration.
Authorization	An SSL proxy can control access to system resources by allowing you to configure the proxy in certain ways. For example, you can configure the proxy to insert client certificate information into client requests and then grant access based on that information. Also, for environments that include an LDAP database server, the proxy can grant access to resources by querying the LDAP server using client certificate credentials.
Network traffic control	A powerful feature of the SSL proxy is its ability to insert different types of headers into client requests. This feature, when used with rule building, allows the proxy to intelligently control how SSL traffic is handled. For example, you can implement persistence for SSL connections by inserting session IDs or client certificate hashes as headers into client requests. Another traffic control feature is the proxy's ability to rewrite HTTP redirections to other servers.
Other SSL features	In addition to the features listed above, the SSL proxy allows you to configure other options such as invalid protocol versions, the size and timeout values of the SSL session cache, and SSL shutdown behavior.

Table 7.1 *Summary of SSL Accelerator proxy features*

Basic configurations

An SSL proxy can be deployed in either of two basic configurations.

- **Client-side SSL proxy**
  A ***client-side SSL proxy*** terminates SSL connections, decrypts a request, and sends the request in clear text to a web server. The proxy then

retrieves a clear-text response (such as a web page) and encrypts the request, before sending the web page back to the client. During the process of terminating an SSL connection, the proxy performs all of the certificate verification functions normally handled by the target web server, as well as encryption and decryption functions. For information on creating a basic client-side SSL proxy, see *Creating a client-side-only SSL proxy*, on page 7-3.

- **Client-side SSL proxy with SSL-to-Server enabled**
  Known as SSL-to-Server, the ***server-side proxy*** option ensures security by allowing the proxy to re-encrypt requests before sending them on to a web server. In addition to this re-encryption, the proxy performs the same validation and verification functions for server certificates that it does for client certificates. For information on enabling the SSL-to-Server feature, see *Creating a client-side proxy with SSL-to-Server enabled*, on page 7-6.

The following sections describe how to create these two different proxy configurations.

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 sections describe these two basic configurations and the procedures for implementing them.

Creating a client-side-only SSL proxy

A client-side proxy performs some basic functions that are automatically configured when you create the proxy. These functions are required, and the BIG-IP system creates default values for those functions. As a proxy administrator, you are free to modify those default values, as a way to customize the way that the proxy performs these basic functions.

Other client-side proxy functions, however, are strictly optional, providing you with a way to even further customize the way that the proxy manages SSL traffic.

The following sections describe these basic and optional functions. Following these sections are the procedures for creating a basic client-side proxy.

Basic client-side functions

The basic functions of a client-side proxy are to off load certificate validation and verification tasks, as well as encryption and decryption, from your target web servers. The proxy does this in the following way:

- First, a client-side proxy attempts to validate and verify any client certificate presented with the request.
- If validation and verification succeeds, the proxy decrypts the request, sends it to a target server as plain text, and waits for a response.
- On receiving the unencrypted response, the proxy encrypts the response and sends it back to the client that originated the request.

In performing these functions, the proxy can use either a set of default values, or a set of new values that you specify.

For more information on certificate verification and encryption and decryption of requests and responses, see *Certificate verification*, on page 7-11, and *Encryption and decryption*, on page 7-39.

Other functions

Additional client-side configuration options are available to further offload tasks from your web servers to the proxy. Examples are the ability for the proxy to rewrite HTTPS redirections from one web server to another, and the ability to maintain persistence for SSL connections.

In addition to off loading work from web servers, a client-side proxy can be configured to intelligently control SSL traffic, either for access control (authorization) purposes, or for load balancing purposes.

For more information on these functions, see *Authorization*, on page 7-42, *Network traffic control*, on page 7-52, and *Other SSL protocol options*, on page 7-63.

Configuration procedures

Prior to creating a client-side SSL proxy, you must first generate a valid x509 certificate from a Trusted Certificate Authority, or generate a temporary certificate, and install it on the BIG-IP system. The BIG-IP system then uses this certificate when acting as a server to receive requests from clients. For more information on proxy certificates, see *Signed certificates*, on page 7-11. For instructions on how to generate and install a certificate on the BIG-IP system, see *Using the Key Management System*, on page 7-29.

Once you have installed a certificate on the BIG-IP system and created a basic client-side SSL proxy, you can configure additional options. For information on configuring additional options, see the remaining sections of this chapter.

◆ Note

Use the following procedure only when creating a client-side proxy with no server-side proxy option enabled. To create a client-side proxy that includes the server-side proxy option, use the procedure described in the section titled ***Creating a client-side proxy with SSL-to-Server enabled****, on page 7-6.*

You can create a client-side proxy using either the Configuration utility or the command line.

To create a client-side 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** box, check the box labeled **SSL.**
4. In the **Proxy Address** box, type the IP address of the proxy.
5. In the **Proxy Service** box, type **https** or select the **https** service from the list box.
6. In the **Destination Address** box, type the IP address of either a local virtual server or a target web server.
7. In the **Destination Target** box, use the **Local Virtual Server** option in the list box, or select **External Node**.
   When you use the default **Local Virtual Server** option, the destination of a packet is a virtual server on the proxy. When you select **External Node**, packets pass through the proxy to a target web server.
8. In the **SSL Certificate** box, select a certificate file name from the list, or type a certificate file name.
9. In the **SSL Key** box, select a key file name from the list, or type a key file name.
10. Click **Done**.

To create a client-side SSL proxy from the command line

Use the following command-line syntax to create a client-side 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 7.2.

```
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 7.2 *Sample SSL proxy entries in* ***/config/bigip.conf*** *with client-side certificate and key files configured*

Creating a client-side proxy with SSL-to-Server enabled

Once the SSL Accelerator proxy has decrypted a client request, you might want the BIG-IP system 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. You implement this feature when you create the client-side SSL proxy.

The following sections describe the basic functions of the server-side proxy, and describe how to create a proxy with the SSL-to-Server feature enabled.

Basic server-side functions

The proxy performs these additional tasks when the SSL-to-Server option is enabled:

- Re-encrypting a decrypted client request and sending it to the server.
- Verifying the server certificate (if required).
- Sending a public key to the server for the purpose of encrypting the server response.
- Decrypting the server response.
- Re-encrypting the response and sending it to the client.

Configuration procedures

Prior to implementing a client-side proxy with the SSL-to-Server feature enabled, you must first generate a valid x509 certificate from a Trusted Certificate Authority, or generate a temporary certificate, and install it on the BIG-IP system. The proxy then uses this certificate when acting as a

server to receive requests from clients. As an option, you can generate and install a second certificate on the BIG-IP system, when SSL-to-Server is enabled and the target server requires the proxy to present client credentials.

◆ Note

If you do not install the second certificate on the proxy for implementing server-side SSL processing (SSL-to-Server), and the target server subsequently requires that the BIG-IP system present client credentials, any server-side connection will fail, causing the corresponding client-side connection to close.

For more information on proxy certificates, see *Signed certificates*, on page 7-11. For instructions on how to generate and install a certificate on the BIG-IP system, see *Using the Key Management System*, on page 7-29.

Once you have installed certificates on the BIG-IP system and created a basic client-side SSL proxy with SSL-to-Server enabled, you can configure additional options. For information on configuring additional options, see the remaining sections of this chapter.

◆ Note

Use following procedure only when creating a client-side proxy with the server-side proxy option enabled. To create a client-side proxy without the server-side proxy option, use the procedure described in the section titled ***Creating a client-side-only SSL proxy****, on page 7-3.*

You can create a client-side proxy with SSL-to-Server enabled using either the Configuration utility or the command line interface.

To create a client-side proxy with 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. In the **Proxy Address** box, type the IP address of the proxy.
5. In the **Proxy Service** box, type **https** or select the **https** service from the list box.
6. In the **Destination Address** box, type the IP address of either a local virtual server or a target web server.
7. In the **Destination Target** box, use the **Local Virtual Server** option in the list box, or select **External Node**.
   When you use the default **Local Virtual Server** option, the

destination of a packet is a virtual server on the proxy. When you select **External Node**, packets pass through the proxy to a target web server.

8. In the boxes labeled **SSL Key** and **SSL Certificate,** either type the names of the key and certificate files or select the names from a list of available key and certificate files.
   These certificate and key files are used to authenticate a proxy acting as a server to clients during client-side SSL processing.
9. Optionally, in the boxes labeled **Server SSL Key** and **Server SSL Certificate**, either type the names of the key and certificate files or select the names from a list of available key and certificate files. When the target server requires the proxy to present client credentials, these certificate and key files are used to authenticate the proxy.
10. Click **Done**.

To create a client-side proxy with SSL-to-Server from the command line

Use the following command line syntax to create a client-side SSL proxy with SSL-to-Server enabled:

```
b proxy <ip>:<service> [unit <unit_id>] \
target <server|virtual> <ip>:<service> \
clientssl enable \
clientssl key <clientssl_key> \
clientssl cert <clientssl_cert> \
serverssl enable \
serverssl key <serverssl_key> \
serverssl cert <serverssl_cert> \
```

For example:

```
b proxy 10.1.1.1:443 unit 1\
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 7.3 shows the state of the **/config/bigip.conf** file, after creating a client-side 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 7.3 *Sample SSL proxy entries in* ***/config/bigip.conf*** *with both client-side and server-side certificate and key files configured*

Displaying SSL Accelerator proxy information

You can use the display information about an SSL proxy by using either the Configuration utility or the **bigpipe proxy** command.

To display configuration information for a proxy using the Configuration utility

1. In the navigation pane, click **Proxies**.
   The Proxies screen opens.
2. Click an existing proxy name.
   The properties page for that proxy is displayed.

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
```

Disabling or deleting an SSL Accelerator proxy

The following procedures show how to disable or delete an SSL proxy.

To disable an SSL proxy using the Configuration utility

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

2. In the Proxy Server column, locate the proxy to be disabled.
3. In the Enabled column, click the button.
   All proxies are enabled by default.

To delete an SSL proxy using the Configuration utility

1. In the navigation pane, click **Proxies**.
   The Proxies screen opens.
2. In the Proxy Server column, locate the proxy to be deleted.
3. In the Delete column, click the Delete button.

To disable or delete an SSL 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**, 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**, type the following command:

```
b proxy 209.100.19.22:443 delete
```

Authentication

Authentication is the process of verifying the identity of a client or server and determining whether or not that client or server can be trusted. When attempting to authenticate a client or server, the proxy first checks the validity of the certificate being presented. If the certificate is valid, the proxy then seeks to verify the certificate, to ensure that the sender can be trusted. The proxy can also check a Certificate Revocation List (CRL), to see if a certificate being presented by a client or server has been revoked.

To help you configure the proxy to handle these various authentication tasks, this section addresses the following topics:

- Certificate verification
- Certificate revocation
- The Key Management System (KMS)

Certificate verification

Certificate verification is the process of determining whether a client or server can trust a certificate that is presented by a peer (that is, a client or a server). When receiving a request from a client or a response from a server, an SSL proxy attempts to verify that the certificate presented by the client or server can be trusted.

Signed certificates

A basic security requirement for clients and servers handling SSL connections is that they each present a certificate signed by a trusted Certificate Authority (CA), whenever they communicate with a peer. Thus, in a client-server scenario where the server requires the presentation of a client certificate, the client presents a certificate to the server servicing the request, and a server presents a certificate to the client receiving the response.

When an SSL proxy is added to the scenario, the BIG-IP system, too, must hold certificates that it can present to clients and servers when processing requests and responses. Specifically, a client-side proxy that requires the presentation of a certificate from a client needs to hold a certificate, to be used when acting as a server to communicate with that client. Optionally, if the SSL-to-Server feature is enabled and a target server requires the presentation of a certificate from the proxy, the proxy can hold a second certificate, to be used when acting as a client to communicate with that target server.

You must generate and install certificates onto the BIG-IP system prior to creating the proxy. You can do this either through the Configuration utility or the **genkey** and **genconf** command-line utilities. As an alternative, you can import existing key/certificate pairs. For more information on importing existing key/certificate pairs, see *Importing keys and certificates*, on page 7-35.

Figure 7.4 shows the distribution of certificates in a client-server configuration in which a client-side proxy with SSL-to-Server enabled has been added.

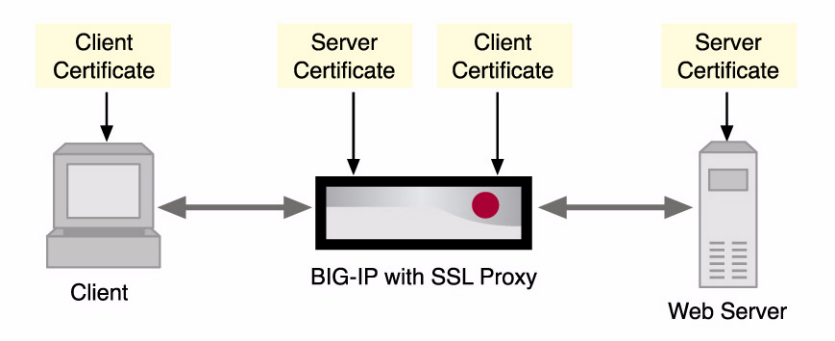

Figure 7.4 *Certificates in a BIG-IP system configuration with an SSL proxy*

The following section provides an introduction to certificate verification as handled by the SSL proxy, and then provides detailed procedures for configuring both client-side and server-side verification.

Summary of certificate verification methods and options

The primary way that a proxy verifies a client or server certificate is through the use of trusted certificate authority (CA) files and Trusted CAs paths. The proxy uses these files and paths to determine whether it should trust a certificate that a client or server presents to it. Moreover, for client-side SSL connections, the SSL proxy can advertise to a client a list of those CAs that the proxy wants the client to recognize as trusted CAs. Finally, the proxy can send a chain file to a client or server, to ensure that the client or server can authenticate the proxy. These various files and paths are described as follows.

- **Trusted CAs files**
  A Trusted CAs file is the primary means by which an SSL proxy verifies a certificate, either from a client or a server. The Client Trusted CAs file and the Server Trusted CAs file each contain a list of Certificate Authorities (CA) that the proxy trusts. When a client or server presents a certificate to the proxy, the proxy checks the certificate against the appropriate list of trusted CAs. If a match exists, then the SSL proxy trusts that client or server.
- **Trusted CAs paths**
  A Trusted CAs path is a secondary means by which a proxy verifies a certificate, either from a client or server. The Client Trusted CAs path and the Server Trusted CAs path contain individual files that each contain a certificate signed by a CA that the proxy trusts. When the proxy cannot find a Certificate Authority listed in one of the Trusted CAs fileTrusted CAs files, it checks either the Client Trusted CAs path or the Server Trusted CAs path, depending on whether the certificate verification is part of client-side or server-side SSL processing.
- **The Client Certificate CA file**
  Unlike a Trusted CAs file, which is known to and used strictly by the SSL proxy, the Client Certificate CA file is directly advertised to clients, to inform them of the CAs that the proxy wants its clients to recognize as trusted CAs.
- **Certificate chain files.**
  When you create a client-side proxy, you can specify a Client Chain file. The Client Chain file is used by a client to authenticate the proxy during the client certificate verification process. If you enable the SSL-to-Server option for server-side processing, you can specify a second chain file, called the Server Chain file. The Server Chain file is used by a server to authenticate the proxy during the server verification process.

In addition to specifying a list of trusted CAs as a way to verify certificates, you can also configure some other verification options on the SSL proxy. These options include specifying whether the SSL proxy should request, require, or ignore certificates presented to it; configuring the SSL proxy to

authenticate a client either once per SSL session or upon each subsequent reuse of the session; and specifying the number of certificates in a chain that the SSL proxy can traverse before verification fails.

Client-side certificate verification

When a client makes an HTTPS request, the SSL proxy performs the client certificate verification task normally performed by the target web server.

Figure 7.5 shows the interaction between a client and an SSL proxy during the certificate verification process.

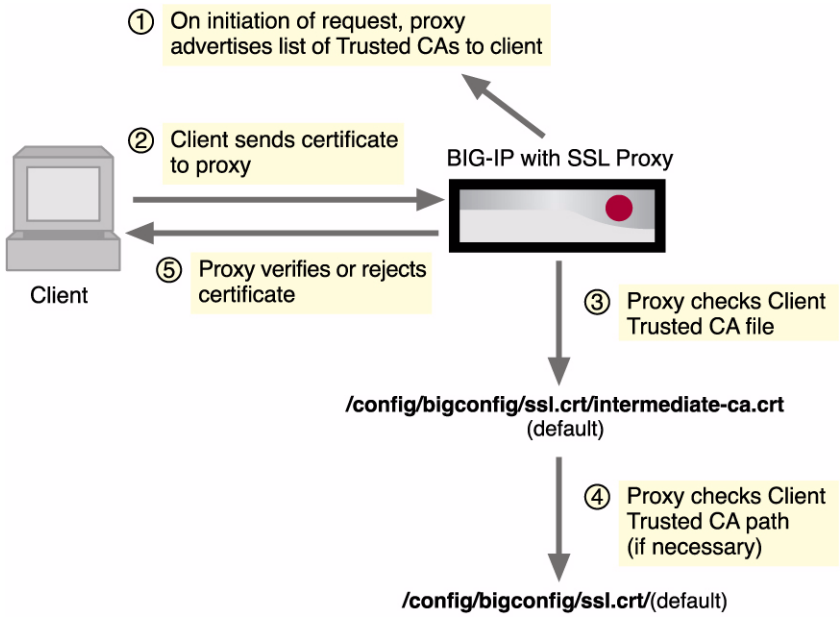

Figure 7.5 *Client-side certificate verification process*

As Figure 7.5 shows, when a client presents a certificate to the proxy, the proxy uses the Client Trusted CAs file or the Client Trusted CAs path to determine the Certificate Authorities that it can trust. Using this file and path is the primary way that a proxy attempts to verify a client certificate. A default Client Trusted CAs file and Client Trusted CAs path are automatically created when you create a client-side proxy. You can either use the default file and path or specify a different file and path. For more information, see *Specifying trusted client CAs*, on page 7-14.

A second factor in client certificate verification is the ***Client Certificate CA file***, which the proxy uses to advertise to clients a list of the CAs that the proxy trusts. Note that this list could be different from the list of CAs that the proxy *actually* trusts. Like the Trusted CAs file and path, a default version of the file is automatically created when you create a client-side proxy. You can either use the default file, or create a different file. For more information, see *Advertising a list of trusted client CAs*, on page 7-16.

Finally, there is a client chain file, which the proxy sends to a client as part of the client certificate verification process. Instead of being used to verify a client certificate, the proxy's client chain file is used by a client to verify the proxy's certificate. The default client chain file is the Client Trusted CAs file. For more information, see *Configuring a client certificate chain*, on page 7-17.

Table 7.2 lists and describes the various options that are available for configuring certificate verification on a client-side SSL proxy

Configuration Option	Description
Specifying trusted client CAs	Allows you to configure certificate chaining and verification.
Advertising a list of trusted client CAs	Allows you to specify the CAs that you would like to advertise to clients as being trusted by the proxy.
Configuring a client certificate chain	Allows you to specify or build a certificate chain file that a client can use to authenticate the proxy.
Configuring the presentation of client certificates	Allows you to configure the SSL proxy to either request, require, or ignore certificates presented by a client.
Configuring per-session authentication	Allows you to specify whether the proxy should authenticate a client once per session or once per session and upon each subsequent reuse of an SSL session.
Authentication depth	Allows you to specify the maximum number of certificates that can be traversed in a client certificate chain.

Table 7.2 *Certificate verification options for a client-side proxy*

The following sections provide detailed descriptions of the options listed above and the procedures for configuring them.

Specifying trusted client CAs

For client-side SSL processing, you can configure the SSL proxy to verify certificates presented by a client. Using either the Configuration utility or the **bigpipe proxy** command, you can specify both a **Client Trusted CAs file** name and a **Client Trusted CAs path** name, which the proxy then uses to verify client certificates. If you do not configure a **Client Trusted CAs file** or **Client Trusted CAs path**, the proxy uses a default file and path.

To configure client-side certificate verification, you might need to configure these elements:

- **The Client Trusted CAs file**
  The ***Client Trusted CAs file*** that you specify for certificate verification contains one or more certificates, in Privacy Enhanced Mail (PEM) format. Built manually, this file contains a list of the client certificates that the SSL proxy will trust. If you do not specify a Client Trusted CAs

file, or the specified Client Trusted CAs file is not accessible to the proxy, the proxy uses the default file name **/config/bigconfig/ssl.default.crt**.

- **The Client Trusted CAs path**
  The ***Client Trusted CAs path*** is a path name to a directory containing the certificates used when searching for trusted CAs. When searching a Client Trusted CAs path, the proxy only examines those certificates that include a properly-formatted symbolic link to a certificate file. If you do not specify a Client Trusted CAs path, or the Client Trusted CAs 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, unlike the Client Trusted CAs file, which can contain more than one certificate.
- **Symbolic links to client certificates**
  Generating symbolic links for certificates ensures that each file in the path is linked to a certificate.

The procedures for specifying a Client Trusted CAs file and a Client Trusted CAs path, and for generating symbolic links to client certificates, follow.

To specify the Client Trusted CAs file, Client Trusted CAs path, and symbolic-link generation using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the boxes **Client Trusted CAs file** and **Client Trusted CAs path**, either select the name of a Trusted CAs file and path from the box, or type the name of a Trusted CAs 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. You should do this whenever you specify any of the path attributes.
5. Click **Done**.

To specify the Client Trusted CAs file and Client Trusted CAs path from the command line

To specify the Client Trusted CAs file and Client Trusted CAs 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>
```

To generate symbolic links for a Client Trusted CAs path from the command line

To generate symbolic links from the command line, you must use the UNIX **make** command.

1. Use the UNIX **cd** command to change to the directory containing the certificates.
2. Run the following command as shown:

   ```
   make -f /config/bigconfig/ssl.crt/Makefile
   ```

3. Restart the SSL proxy by either running the **proxyd** command from the command line or by reconfiguring the proxy in the Configuration utility.

◆ Note

Whenever you subsequently add, remove, or modify certificates, you must restart the SSL proxy again, as described in Step 3 above. This causes the changes to take effect.

Advertising a list of trusted client CAs

If you intend to configure the SSL proxy to require or request client certificates for authentication, you will want the proxy to send to clients a list of CAs that the server is likely to trust.

This list, known as the Client Certificate CA file, is different from the Client Trusted CAs file. This is because, in some cases, you might have a client that does not possess a valid client certificate, in which case you might not want to reveal the actual list of CAs that the proxy trusts. The Client Certificate CA file solves this problem by allowing the proxy to advertise a list of CAs that is different from the actual Client Trusted CAs file configured as part of certificate verification.

◆ Tip

Although the contents of the Client Certificate CA file can differ from that of the Client Trusted CAs file, it is best, for compatibility reasons, to set the Client Certificate CA option to match the actual Client Trusted CAs file. This is because modern borwsers might not permit SSL session negotiation to proceed if the peer that requests the client certificate does not provide a list of trusted CAs.

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 Client Certificate CA 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 use the command line to configure the proxy to send a list of trusted CAs to a client, type the **bigpipe proxy** command, using the following arguments:

```
b proxy <ip>:<service> [clientssl] client cert ca <clientside client cert CA file name>
```

Configuring a client certificate chain

In any client verification process, not only does the proxy need to authenticate the client, but the client might need to authenticate the proxy. However, a certificate that the SSL proxy uses to authenticate itself to a client is sometimes signed by an intermediate CA that is not trusted by that client. In this case, the proxy might need to use a certificate chain. The proxy enables you to specify 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 file you specified has a **.chain** extension.
2. If the file specification does not include a **.chain** extension, the proxy appends that extension to the specified file and then searches for that file.
3. If the file is not found, the proxy instead appends a **.crt** extension to the specified 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 certificate name **www.mysite.com.crt**, replace the **.crt** file name extension with the **.chain** extension, and search on the file name **www.mysite.com.chain**.
5. If unable to find the certificate chain using the preceding procedure, the proxy attempts to build the chain.

To specify 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**, either select the name of a Client Trusted CAs file from the box, or type the name of a Client Trusted CAs file.
4. Click **Done**.

To specify a certificate chain from the command line

To specify 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>
```

Configuring the presentation of client certificates

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. In this case, if certificate verification fails, all corresponding connections are closed, and log messages are generated by the **proxyd** daemon.
- You can configure the proxy to ***ignore*** a certificate (or lack of one) and therefore never authenticate the client. The **ignore** setting is the default setting, and when used, causes any per-session authentication setting to be ignored. For information on configuring per-session authentication, see *Configuring per-session authentication*, on page 7-19.

*If are using the LDAP-based client authorization feature, use of the **request** or **ignore** options can sometimes cause a connection to terminate. For more information, see Creating an authorization model, on page 7-45.*

◆ 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 enables the BIG-IP system or a server to then perform actions such as redirecting the request to another server, sending different content back to the client, or performing client certificate or session ID persistence.*

To configure client-side certificate presentation 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 certificate presentation from the command-line

To configure client-side certificate presentation 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>
```

Configuring per-session authentication

You can configure an SSL proxy to require authentication either once per SSL session (**once**), or once upon each subsequent reuse of an SSL session (**always**). The default setting for this option is **once**.

Whether you set this value to **once** or **always** depends on your application. A well-designed web application should only need to verify a certificate once per session. We recommend for performance reasons that you use the default setting (**once**) whenever possible.

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.
   The **Client Authenticate Once** box is checked.
3. Click on the **Client Authenticate Once** box
   This clears the box, thereby changing 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. This changes the setting from **once** to **always**. The command line syntax for this operation s as follows:

```
bigpipe proxy <ip>:<service> [clientssl] authenticate <once | always>
```

Configuring 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 **9**. If a longer chain is provided, and the client has not been authenticated within this number of traversals, client certificate verification fails. If the authentication depth value is set to zero, then only the client certificate, and one of the chain file, is examined.

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>
```

Server-side certificate verification

Server-side verification occurs when the ***SSL-to-Server*** feature is enabled and the presentation of a server certificate to the proxy is set to **require**. Figure 7.6 shows the interaction between an SSL proxy and a server during the certificate verification process.

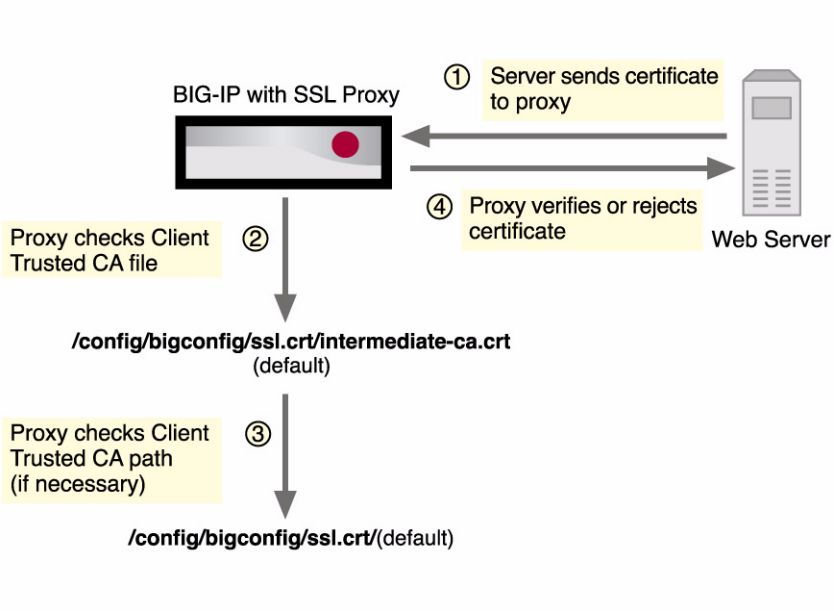

Figure 7.6 *Server-side certificate verification process*

As Figure 7.6 shows, when a server presents a certificate to the proxy, the proxy uses the Server Trusted CAs file or the Server Trusted CAs path to determine which Certificate Authorities it can trust. Using this file and path is the primary way that a proxy attempts to verify a server certificate. A default Server Trusted CAs file and Server Trusted CAs path are automatically created when you enable the SSL-to-Server option. You can either use the default file and path, or specify a different file and path. For more information, see *Specifying trusted server CAs*, on page 7-22.

In addition to the Trusted CAs file and Trusted CAs path, there is a server chain file, which the proxy sends to a server as part of the entire server certificate verification process. Instead of being used to verify a server certificate, however, the proxy's server chain file is used by a server to verify the proxy's certificate. The default server chain file is the Server Trusted CAs file. For more information, see *Configuring a server certificate chain*, on page 7-23.

Table 7.3 lists and describes the various options available for configuring certificate verification on a server-side SSL proxy

Configuration Option	Description
Specifying trusted server CAs	Allows you to configure certificate chaining and verification.
Configuring a server certificate chain	Allows you to specify or build a certificate chain file that a server can use to authenticate the proxy.
Configure presentation of server certificates	Allows you to configure the SSL proxy to either require or ignore certificates presented by a server.
Authentication depth	Allows you to specify the maximum number of certificates that can be traversed in a server certificate chain.

Table 7.3 *Certificate verification options for a server-side proxy*

The following sections provide detailed descriptions of the options listed above and the procedures for configuring them.

Specifying trusted server CAs

For server-side SSL processing, you can configure the SSL proxy to verify certificates presented by a server. Using either the Configuration utility or the **bigpipe proxy** command, you can specify both a Server Trusted CAs file name and a Server Trusted CAs path name, which the proxy then uses to verify server certificates. You can also generate symbolic links to ensure that every file in the path is linked to a certificate.

To configure server-side certificate verification, you might need to configure the following elements:

- **The Server Trusted CAs file**
  The ***Server Trusted CAs file*** that you specify for certificate verification contains one or more certificates, in PEM format. Built manually, this file contains a list of the server certificates that the SSL proxy will trust. If you do not specify a Server Trusted CAs file, or the specified Server Trusted CAs file is not accessible to the proxy, the proxy uses the default file name **/config/bigconfig/ssl.crt/default.crt**.
- **The Server Trusted CAs path**
  The ***Server Trusted CAs path*** is a path name to a directory containing the certificates used when searching for trusted CAs. When searching a Server Trusted CAs path, the proxy only examines those certificates that include a properly-formatted symbolic link to a certificate file. If you do not specify a Server Trusted CAs path, or the Server Trusted CAs 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.

- **Symbolic links to server certificates**
  Generating symbolic links for certificates ensures that each file in the path is linked to a certificate.

The procedures for specifying a Trusted CAs file and a Trusted CAs path, and for generating symbolic links to server certificates follow.

To specify the Server Trusted CAs file, Server Trusted CAs path, and symbolic-link generation using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the boxes **Server Trusted CAs file** and **Server Trusted CAs path,** either select the name of a Trusted CAs file or path from the box, or type the name of a Trusted CAs 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 Server Trusted CAs Path** check box. You should do this whenever you specify any of the path attributes.
5. Click **Done**.

To specify the Server Trusted CAs file and Server Trusted CAs path from the command line

To specify the Server Trusted CAs file and Server Trusted CAs path from the command line, type the **bigpipe proxy** command, using the appropriate arguments, as follows:

```
b proxy <ip>:<service> serverssl ca file <serverside CA file name>
b proxy <ip>:<service> serverssl ca path <serverside CA path name>
```

To generate symbolic links for a server CRL path

For the procedure on generating symbolic links for a Server Trusted CAs path, use the procedure described in *To generate symbolic links for a Client Trusted CAs path from the command line*, on page 7-16.

Configuring a server certificate chain

In any server verification process, not only does the proxy need to authenticate the server, but the server might need to authenticate the proxy. However, a certificate that the SSL proxy uses to authenticate itself to a server is sometimes signed by an intermediate CA that is not trusted by that server. In this case, the proxy might need to build a certificate chain.

You can 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 optionally-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.

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 **Server Chain File**, either select the name of a Server Trusted CAs file from the box, or type the name of a Server Trusted CAs 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> serverssl chain <serverside chain file name>
```

Configuring the presentation of server certificates

To implement certificate verification on the server side (that is, between the SSL proxy and the server), you can configure the proxy either to require the server to present a certificate or to ignore the presentation of a certificate. Note, however, that you cannot require the server to present a certificate if anonymous cipher suites are negotiated. For information on configuring ciphers, see *Specifying SSL ciphers*, on page 7-40.

If this option is set to **require**, the proxy attempts to verify 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** (the default setting), 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

Specify this option as **serverssl server cert** on the **bigpipe proxy** command line. The following command shows an example:

```
b proxy <ip>:<service> serverssl server cert require
```

Authentication depth

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
```

Certificate revocation

The SSL proxy feature includes support for certificate revocation lists (CRLs). This CRL support is in the form of a CRL file and a CRL path. Like the Trusted CAs files and paths, the proxy enables you to configure one CRL file and path for the client-side proxy, and one CRL file and path for the server-side proxy.

When specifying a CRL path (client-side or server-side), you can also generate symbolic links for each CRL.

Important

CRL files can become outdated, and might need to be updated as often as every day, or as seldom as every 30 days. If your CRL file is out-of-date, the BIG-IP system rejects ***all*** *certificates, both valid and invalid. For this reason, it is important to keep your CRL files up-to-date at all times. You can do this by issuing the following command from the* ***/config/bigconfig/ssl.crl*** *directory:* ***openssl crl -in <crlname> -text -noout****.*

Client-side certificate revocation

To configure CRLs for a client-side proxy, you might need to configure the following elements:

- **The Client CRL file**
  The Client CRL file contains a list of revoked client certificates. When specifying a list of revoked client certificates, the file that you specify must be a PEM-formatted file.
- **The Client CRL path**
  The path name you specify is the path to a directory with CRLs and corresponding symbolic links. If the CRL path is not specified or accessible, the proxy uses the path **/config/bigconfig/ssl.crl/** by default.
- **Symbolic links to revoked certificates**
  Generating symbolic links for CRLs ensures that each file in the path is linked to a revoked certificate. Once these links are generated, authentication fails if a client presents a revoked certificate as part of the client authentication process.

The procedures for specifying a client CRL file and a client CRL path, and for generating symbolic links to revoked client certificates follow.

To specify the Client CRL file, Client CRL path, and symbolic-link generation using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the boxes **Client CRL File** and **Client CRL Path**, either select the name of a CRL file or path from the box, or type the name of a CRL 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 CRL Path** check box. You should do this whenever you specify any of the path attributes.
5. Click **Done**.

To specify the Client CRL file and Client CRL path from the command line

To specify the Client CRL file and Client CRL path from the command line, type the **bigpipe proxy** command, using the appropriate arguments, as follows. Note that this command does not generate the symbolic links necessary for the CRL path. For information on generating symbolic links from the command line, see the next procedure.

```
b proxy <ip>:<service> [clientssl] crl file <clientside crl file name>
b proxy <ip>:<service> [clientssl] crl path <clientside crl path name>
```

To generate symbolic links for a client CRL path from the command line

To generate symbolic links from the command line, you must use the UNIX **make** command. The procedure is as follows.

1. Use the UNIX **cd** command to change to the directory containing the CRLs.
2. Run the following command as shown:

   ```
   make -f /config/bigconfig/ssl.crl/Makefile
   ```

3. Restart the SSL proxy by either running the **proxyd** command from the command line or by reconfiguring the proxy in the Configuration utility.

◆ Note

*Whenever you subsequently add, remove, or modify CRLs, you must run the **make** command again (only if a client CRL path is specified) and restart the SSL proxy again, as described in Steps 2 and 3 above. This causes the changes to take effect.*

Server-side certificate revocation

To configure CRLs for a server-side proxy, you might need to configure the following elements:

- **The Server CRL file**
  The Server CRL file contains a list of revoked server certificates. When specifying a list of revoked server certificates, the file that you specify must be a PEM-formatted file.
- **The Server CRL path**
  The path name you specify is the path to a directory with CRLs and corresponding symbolic links. If the CRL path is not specified or accessible, the proxy uses the path **/config/bigconfig/ssl.crl/** by default.

- **Symbolic links to revoked certificates**
  Generating symbolic links for CRLs ensures that each file in the path is linked to a revoked certificate. Once these links are generated, authentication will fail if a server presents a revoked certificate as part of the server authentication process.

The procedures for specifying a server CRL file or a server CRL path, and for generating symbolic links to revoked server certificates follow.

To specify the Server CRL file and Server CRL path using the Configuration utility

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the boxes **Server CRL File** and **Server CRL Path,** either select the name of a CRL file or path from the box, or type the name of a CRL 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 Server CRL Path** check box. You should do this whenever you specify any of the path attributes.
5. Click **Done**.

To specify the Server CRL file and Server CRL path from the command line

To specify the Server CRL file and Server CRL path from the command line, type the **bigpipe proxy** command, using the appropriate arguments, as follows:

```
b proxy <ip>:<service> serverssl crl file <serverside crl file name>
b proxy <ip>:<service> serverssl crl path <serverside crl path name>
```

To generate symbolic links for a server CRL path

For the procedure on generating symbolic links for a server CRL path, use the procedure described in *To generate symbolic links for a Client Trusted CAs path from the command line*, on page 7-16.

Using the Key Management System

Managing certificates and public-private key pairs can be confusing, especially when you have more than one proxy defined. To help you manage certificates and keys, the BIG-IP system provides a set of key management screens within the Configuration utility.

Collectively, these screens are known as the ***Key Management System (KMS)***.

The Configuration utility includes a set of screens for managing key pairs and certificates. Available under the Cert Admin tab on the main Proxies screen, these screens allow you to:

- List information about all existing key pairs and certificates
- Generate new key pairs and certificate requests
- Regenerating certificate requests
- Install certificates for existing key pairs
- Associate certificates with proxies
- Displaying a key, certificate, and SSL proxy properties
- Import and export PEM-formatted keys and certificates
- Import and export non-PEM-formatted keys and certificates (packaged archives only)

The following sections describe these features and how to use them.

Administering keys and certificates

Summary information about existing key pairs and certificates is available on the SSL Certificate Administration screen, available from the main Proxies screen. Figure 7.7 shows an example of the SSL Certificate Administration screen.

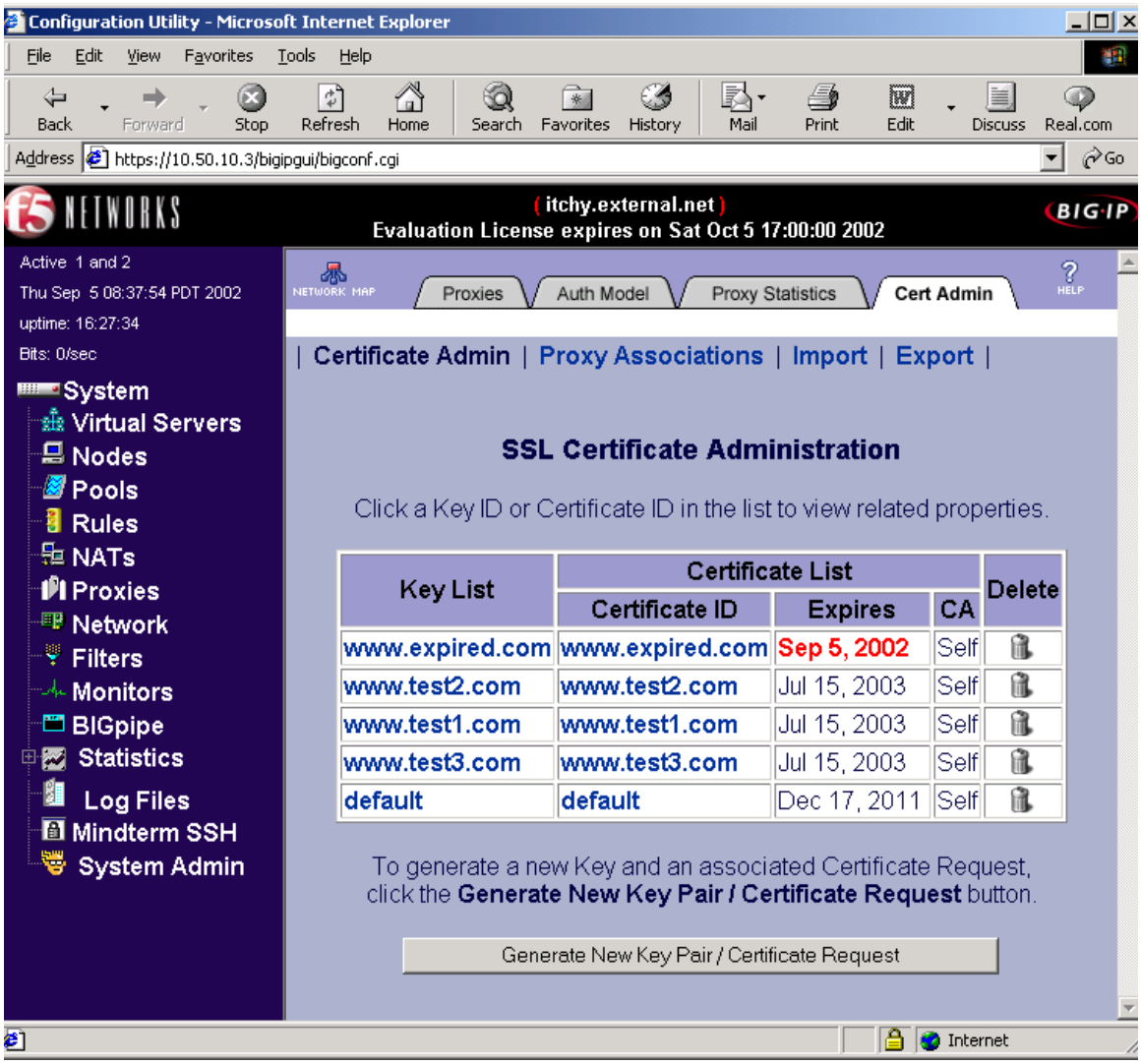

Figure 7.7 *The SSL Certificate Administration screen*

The SSL Certificate Administration screen displays the following information:

Key List
The list of identification names associated with existing key pairs. Note that only the first 20 characters of a key name appear in this box. To see a key name that exceeds 20 characters, click on the key name and view its properties.

Security Type
The type of key. Possible values are **NONE** and **FIPS**. This column only appears if you have the FIPS-140 Accelerator option installed.

Certificate ID
The identification name of the certificate associated with the key pair. Note that only the first 20 characters of a certificate ID appear in this box. To see a certificate ID that exceeds 20 characters, click on the certificate ID and view its properties.

Expires
The date on which the certificate expires. Expired dates appear in red.

CA
The certificate authority that issued the certificate. Self-signed certificates, (where the issuer and the subject are the same), are indicated as **Self**.

Generate
A button that displays the Generate Certificate Request screen, used to generate a certificate for the key pair shown. This button only appears when no certificate request has been generated for the key shown.

Install
A button that displays the Install Certificate screen, used to install a certificate for the key pair shown. This button only appears when no certificate is installed for the key shown.

Generate New Key Pair/Certificate Request
A button that displays the Generate New Key Pair/Certificate Request screen, used to generate a new key pair and request a certificate for that key pair.

The following procedures describe how to use the SSL Certificate Administration screen.

To list key pair and certificate information

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. When finished, click **Done**.

Generating keys and certificate requests

Using the SSL Certificate Administration screen within the Configuration utility, you can either generate a certificate request for an existing key pair, or you can generate a new key pair and certificate request.

To generate a certificate request for an existing key pair

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. In the **Key List** column, locate the key pair for which you want to generate a certificate request.
4. In the **Certificate ID** column, click **Generate**.
   This displays the Create Certificate Request screen.
5. Type in the data for all boxes on the screen.
6. Click the **Generate Certificate Request** button.

To generate a new key pair and certificate request

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. Click the **Generate New Key Pair/Certificate Request** button.
   This displays the Create Certificate Request screen.
4. Type in the data for all boxes on the screen.
5. Click the **Generate Key Pair/Certificate Request** button.

Regenerating certificate requests

If you already have a certificate for an existing key pair, and want to regenerate that certificate, use the following procedure.

To regenerate a certificate for an existing key pair

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. In the Certificate ID column, click on the certificate that you want to regenerate.
   The Certificate Properties screen displays.
4. Click the **Request Certificate** button.

Installing certificates

Using the SSL Certificate Administration screen within the Configuration utility, you can install a certificate for an existing key pair. Also, if you have regenerated a certificate for an existing key pair (see *Regenerating*

certificate requests, on page 7-32), you can install that certificate, using the Certificate Properties screen. To perform these tasks, use the following procedures.

To install a certificate for an existing key pair

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. In the Key List column, locate the key pair for which you want to install a certificate.
4. In the Certificate ID column, click **Install**.
   This displays the Install SSL Certificate screen.
5. Select either Option 1 or Option 2 and follow the instructions on the screen.
6. In the **Certificate Identifier** box, type a name for the certificate, or choose a name from the list. This name constitutes the certificate ID.
7. Click the **Install Certificate** button**.**

To install a regenerated certificate

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. In the Certificate ID column, click on the certificate that you regenerated.
   The Certificate Properties screen displays.
4. Click the **Install Certificate** button.

Displaying properties

You can display the properties of a certificate or an SSL proxy, using the following procedures.

To display certificate properties

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. In the Certificate ID column, click the certificate ID for which you want to view properties.
   This displays all properties of that certificate.
4. Click **Return to Certificate Administration**.

To display SSL proxy properties

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. Click the Proxy Associations tab.
   This displays the SSL Certificate Proxy Associations screen.
4. Select the proxy for which you want to view properties.
   This displays all properties of the selected proxy.
5. Click **Return to Certificate Administration**.

Deleting keys and certificates

You can delete keys and certificates, using the following procedure.

To delete keys and certificates

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. In the Key List column, identify the keys and certificates you want to delete.
4. If a trashcan icon appears in the Delete column, click the trashcan. This deletes the keys and certificate. If no trashcan icon appears in the Delete column, you cannot delete the keys and certificates because they are associated with a proxy.

◆ Note

A trashcan icon does not appear in the Delete column for a key pair and certificate if the associated SSL proxy is currently in use.

Binding a certificate to a key

Sometimes, a certificate ID does not match the key ID of the key used to create the certificate. This causes the certificate to lack an associated key. For example, certificate **xyz** might have been created using key **abc**. In this case, the certificate **xyz** is not bound to key **abc**. You can correct this problem by binding the certificate to the key.

To bind a certificate to a key

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. In the **Certificate ID** column, click the name of the certificate that you want to bind to a key.
   This displays all properties of that certificate.

4. In the **Associated Key** box, choose the key to which you want to bind the certificate.
5. Click **Apply**.

Associating certificates with SSL proxies

Once you have generated a key pair and installed a certificate for that key pair, you can use the **Proxy Associations** screen to add a proxy for an existing certificate. You can also use this screen to view the properties of an existing proxy.

To add a proxy for an existing certificate

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. Click the Proxy Associations tab.
   This shows all existing key pairs and certificates and the proxies to which they are assigned, if any.
4. Identify the keys and certificate that you want to associate with a new proxy.
5. In the **Add Server** column, click the corresponding **Add** button.
   This opens the Add Proxy wizard, which is the same wizard that is available from the main Proxies screen. Note that if the proxy you are adding is a client-side proxy, the key and certificate fields are automatically filled in on the Add Proxy screen with the names of the corresponding keys and certificate.
6. Enter all pertinent information into the appropriate fields. For detailed information on creating a proxy, see *Creating an SSL Accelerator proxy*, on page 7-3.
7. Click **Done**.

Importing keys and certificates

With the Import tab, you can import either an exported key pair, certificate, or a key/certificate archive into the Key Management System. This screen can only be used when the certificate you are importing is in Privacy Enhanced Mail (PEM) format.

To import a key pair

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. Click the Import tab.
   This displays the SSL Key/Certificate/Archive Import screen.
4. In the **Import Type** box, choose the **Key** button.

5. Click **Continue**.
   This displays the Install SSL Key screen.
6. Follow the instructions on the screen to install the key that you want to import.
7. Click **Install Key**.
   This installs the key and displays its properties.

To import a certificate

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. Click the Import tab.
   This displays the SSL Key/Certificate/Archive Import screen.
4. In the **Import Type** box, click the **Certificate** button.
5. Click **Continue**.
   This displays the Install SSL Certificate screen.
6. Follow the instructions on the screen to install the certificate that you want to import.
7. Click **Install Certificate**.
   This installs the certificate and displays its properties.

To import a key/certificate archive

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. Click the Import tab.
   This displays the SSL Key/Certificate/Archive Import screen.
4. In the **Import Type** box, choose the **Archive** button.
5. Click **Continue**.
   This displays the Choose SSL Key/Certificate Archive screen.
6. Type in an archive file name or use the **Browse** button to select it.
7. Click **Continue**
   This displays the Import Archive screen.
8. Using the left and right arrows on the screen, select the keys and certificates from the archive that you would like to import.
9. Click **Install Included Items**.

Exporting keys and certificates

The Export screen enables you to export a key/certificate archive from the key management system for the purpose of transferring them to another SSL proxy on the network. Once you have exported the selected keys, certificates, or key pair/certificate archives, you can use the KMS Import function to download them to the target SSL proxy.

The Export screen can only be used when the certificate you are exporting is in Privacy Enhanced Mail (PEM) format. Therefore, you cannot export FIPS keys, that is, keys generated on a FIPS-140 system.

To export a key pair

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. Click the Export tab.
   This displays the SSL Key/Certificate/Archive Import screen.
4. In the **Export Type** box, click the **Key** button.
5. From the list box, select a key pair name.
6. Click **Continue**.
7. Follow the instructions on the screen.

To export a certificate

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. Click the Export tab.
   This displays the SSL Key/Certificate/Archive Import screen.
4. In the **Export Type** box, click the **Certificate** button.
5. From the list box, select a certificate name.
6. Click **Continue**.
7. Follow the instructions on the screen.

To export a key/certificate archive

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. Click the Export tab.
4. In the **Export Type** box, click the **Archive** button.
5. From the list box, select an archive name.
6. Click **Continue**.

7. Using the left and right arrows on the screen, select the keys and certificates that you would like to include in the archive.
8. Click **Download Archive**.

Converting keys on a FIPS-140 system

If you have a FIPS system, and that system has non-FIPS keys installed on it, you can perform a one-way conversion of those keys from non-FIPS to FIPS.

To convert non-FIPS keys to FIPS keys

1. In the navigation pane, click **Proxies**.
2. Click the Cert Admin tab.
3. In the **Key List** column, click the key that you want to convert. This displays all properties of that key.
4. In the **Type** box, select **FIPS**.
5. Click **Apply**.
6. When prompted, confirm the one-way conversion action.

Storing private keys securely

Private keys are traditionally stored as plain text on hard disks, where they are susceptible to tampering or theft. For SSL proxy keys, however, you can avoid this risk by using the keys of a FIPS-140 Hardware Security Module (HSM). Such keys conform to government-approved security standards. For more information, see ***Platform Guide: 520/540***.

◆ Note

Multiple-vendor HSMs are incompatible within the same BIG-IP system.

Encryption and decryption

One of the functions of the SSL proxy is to handle encryption and decryption tasks that are normally performed by a web server as part of processing a client request. When configured as a client-side-only proxy, the proxy decrypts incoming requests before sending them on in plain text to the target server. When the SSL-to-Server feature is enabled, the proxy provides an additional level of security by re-encrypting the request before sending it on to the target server.

Figure 7.8 shows how a client-side proxy encrypts and decrypts application data as it passes from client to server and back again.

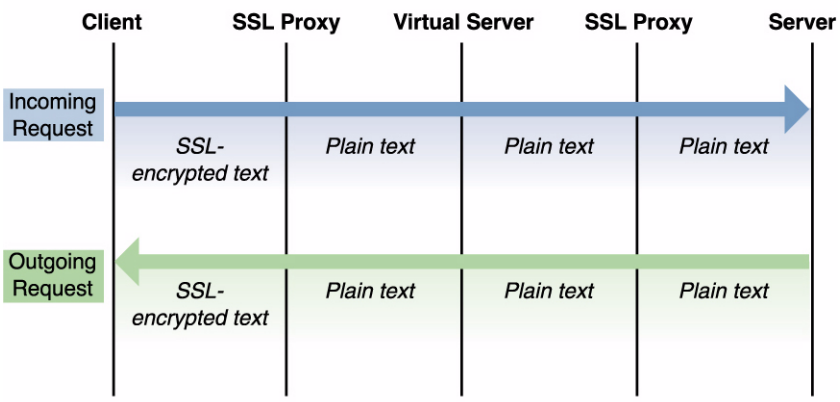

Figure 7.8 *Encryption and decryption with client-side proxy*

Figure 7.9 shows how a client-side proxy with SSL-to-Server enabled encrypts and decrypts application data as it travels on the same route as the packet shown in Figure 7.8.

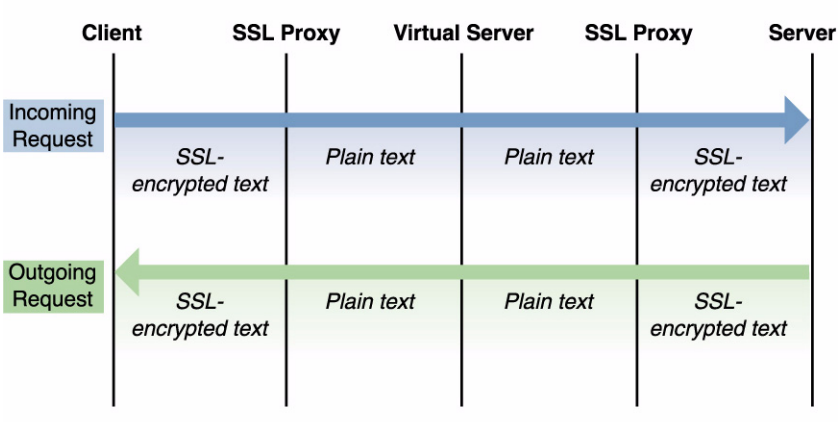

Figure 7.9 *Encryption and decryption with SSL-to-Server enabled*

In addition to creating a client-side proxy and enabling the SSL-to-Server feature, you can configure other encryption options. Table 7.4 summarizes these options.

Configuration Option	Description
Specifying SSL ciphers	Allows you to specify to client and servers the specific ciphers the proxy can support.
Inserting cipher specifications into HTTP requests	Allows you to insert cipher specifications as headers into HTTP requests and then direct the connection based on the cipher specified.

Table 7.4 *Options for configuring encryption on an SSL proxy*

The following sections describe these options and the procedures for configuring them.

Specifying SSL ciphers

For each SSL proxy, you can specify the ciphers available for SSL connections.

When configuring ciphers, you must ensure that the ciphers configured for the SSL proxy match those of the client sending a request, or a server sending a response.

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 fails because there are no common ciphers enabled. This results in the client connection being closed. If the client is using a browser, the user is likely to receive an error message indicating that the web page failed to load.

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, or you can use the default cipher string, **DEFAULT**. The **DEFAULT** cipher string is normally defined as **ALL:!ADH:RC4+RSA:+SSLv2:@STRENGTH.**

Use of the **SSLv2** cipher is not recommended unless necessary. Therefore, concerned users should set the cipher string to **DEFAULT:-SSLv2**.

Note

The SSL proxy supports any combination of ciphers supported by OpenSSL version 0.9.6, except for the IDEA cipher suite. To see the complete syntax for the cipher list, see this 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, using the following syntax:

```
b proxy <ip>:<service> [clientssl] ciphers \"quoted string\"
b proxy <ip>:<service> serverssl ciphers \"quoted string\"
```

Note that you can use the **openssl ciphers** command to test the validity of a cipher string. The syntax for this command is as follows:

```
openssl ciphers -v [cipherlist]
```

For example, the following command lists all OpenSSL ciphers, including NULL ciphers:

```
openssl ciphers -v 'ALL:eNULL'
```

Inserting cipher specifications into HTTP requests

The SSL proxy includes a powerful feature for managing your SSL traffic. This feature gives you the ability to insert headers into incoming requests, and then to direct the traffic based on the information in those headers.

One of the header types that the proxy can insert into an HTTP request is a specification of the cipher that the client used to encrypt its request. After configuring the proxy to insert this information into a request, you can then build a rule that reads the header, through use of the **http_header** variable, and directs the request to a specific pool of servers.

For more information on using this feature to control the destination of client requests, see *Inserting headers into HTTP requests*, on page 7-52.

Authorization

Unlike authentication, authorization is not about trusting identity, but about controlling access to system resources. Once the SSL proxy has verified that a client or server can be trusted, the proxy can then control the connection's level and type of access to the destination content.

The SSL proxy configuration options that support access control for SSL connections are:

- Inserting client certificate fields into HTTP requests
- Limiting the number of concurrent client TCP connections
- Client authorization with an LDAP database server

The following sections describe these authorization options.

Inserting client certificate fields into HTTP requests

One of the most useful ways to control a client's access to system resources is to configure a proxy to insert fields of a client certificate as headers into client requests. For example, properly configured, a proxy can insert the status of a client certificate as a header into a request, and then build a rule that uses the **http_header** variable to select the target web server based on that status.

For more information on using this header insertion feature to control the destination of client requests, see *Inserting headers into HTTP requests*, on page 7-52.

Limiting concurrent TCP connections

When configuring the SSL proxy, you can define the maximum number of client TCP connections allowed per proxy.

If you define a value of **0**, the BIG-IP system performs no limit-checking. There is no upper limit on the maximum number of connections allowed.

To specify the maximum number of allowed client TCP connections

1. In the navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Client TCP Connection Limit** box, type a number.
4. Click **Done**.

Configuring LDAP-based client authorization

For environments that include a Lightweight Directory Access Protocol (LDAP) server, an SSL proxy can perform authorization of client requests. Like the proxy authentication functions, part of client authorization is based on certificates. The other part of proxy authorization is based on user groups and roles that you define.

The remainder of this section is organized into the following topics:

- Authorization types
- Authorization models
- Configuring client authorization on an SSL proxy

◆ Note

The LDAP-based client authorization feature depends on external client authentication being configured through the Setup utility.

Authorization types

An ***authorization type*** is the type of database that the SSL proxy uses to store authorization data for client requests. Currently, the proxy supports one authorization type, known as **LDAP.** The name LDAP refers to the Lightweight Directory Access Protocol, on which the client authorization database is based.

Authorization models

Client authorization by an SSL proxy is based on an authorization model that you define. An ***authorization model*** consists of a set of LDAP parameter values that you define for a proxy. The proxy then uses these values when searching the database during client authorization. You can define multiple authorization models, each with a different set of parameter values, but you can assign only one model per proxy.

Figure 7.10 shows sample values for an authorization model as specified on the **bigpipe authz** command line.

```
bigpipe authz authmodel
    method ldap \
     cachetimeout 200 \
    ldap searchtype certmap \
    ldap servers 192.168.10.1:389 192.168.10.2 \
    ldap admindn "cn=admin,dc=f5,dc=com" \
    ldap adminpw "secret" \
    ldap certmap base = "ou=AuthzLDAPCertMap,dc=f5,dc=com" \
    ldap certmap key: "authzLDAPmap" \
    ldap certmap useserial false \
    ldap user key uid \
    ldap group base "dc=f5,dc=com" \
    ldap group key group \
    ldap group member key member \
    ldap valid groups webusers webadmin
bigpipe proxy 192.168.198.67:443 authz models authmodel
           authz set remoteuser hdr enable
```

Figure 7.10 *Sample parameter values for an LDAP-based authorization model*

With an LDAP-based authorization model, a proxy can authorize clients based on signed client certificates issued by trusted CAs. Then, to further enhance the ability of the proxy to authorize client requests, an authorization model can also include parameters that specify groups and roles. Basing authorization on not only certificates but also groups and roles provides the flexibility you need to control client access to system resources.

The following sections describe these two methods of authorization.

Certificate-based authorization

During the process of authorizing a client, the proxy must search the LDAP database. When using certificate-based authorization, the proxy can search the LDAP database in three ways. The following sections describe these three distinct search types.

- **CERT - Comparing incoming certificates to certificates in the LDAP database**
  Many LDAP server environments already incorporate certificates into the user profiles stored in the LDAP database. Therefore, one way of configuring authorization in LDAP server environments is to configure the proxy to compare an incoming certificate to the certificate stored in the LDAP database for the user associated with the client request. If the certificate is found in the user's LDAP profile, access is granted to the user and the request is granted.
- **CERTMAP - Searching for certificates in certificate-to-user mappings**
  If you create an object and class that maps certificates to users in the LDAP database, you can then configure the proxy to search for a certificate in the map, and retrieve a user from that map. The proxy then checks to ensure that the user in the LDAP database is a valid user.

- **USER - Extracting usernames from incoming certificates**
  If certificates are not stored in the LDAP database, you can configure the proxy to extract a user name from the certificate presented as part of the incoming client request. The proxy then checks to see if an entry for the user exists in the LDAP database. This scenario is a good choice for a company that acts as its own Certificate Authority, where the company is assured that if the certificate is verified, then the user is authorized.

Regardless of the type of certificate-based authorization being performed, the process yields the results shown in Table 7.5.

Result of search	Authorization status
No records match	Authorization fails
One record matches	Authorization succeeds and is subject to groups and roles
Two or more records match	Authorization fails, due to invalid database entries

Table 7.5 *Search results and corresponding authorization status*

Group-based and role-based authorization

In addition to enabling certificate-based authorization, you can also configure authorization based on groups and roles.

- **Groups**
  Because LDAP servers already have the concept and structure of groups built into them, the proxy can include groups in its authorization feature. To enable the use of groups for authorization purposes, you must indicate the base and scope under which the proxy will search for groups in the LDAP database. Also, you must specify attribute values for "group" and "member". Once these tasks have been completed, the proxy can search through the list of valid groups until a group is found that has the current user as a member.
- **Roles**
  Unlike a group, a role is an attribute directly associated with a user. Any role-based authorization performed by the proxy depends on the LDAP database having the concept of roles built into it. To determine if a user should be granted access to a resource, the proxy searches through the roles assigned to the user and attempts to match that role to a valid role defined by the administrator.

Creating an authorization model

To fully implement SSL client authorization on the BIG-IP system, you must create an authorization model, consisting of a set of parameter values. Associated with the **ldap** authorization type, these parameters are normally configured, with default values, either through the Configuration utility or

through the **bigpipe proxy** command. Table 7.6 lists and describes the configurable authorization parameters for an LDAP-based authorization model.

Authorization model parameters	Description
LDAP Basic Parameters	
Method	The type of database being used for authorization. Currently, LDAP is the only method that can be specified.
Cache timeout	The length of time in seconds that items are kept in the authorization cache. Note that Configuration utility and the command line interface do no valid range-checking.
LDAP General Parameters	
Server list	The list of IP addresses and port numbers for all of the potential LDAP servers.
Secure	A parameter that instructs the BIG-IP system to use secure LDAP communication (that is secure communication to the LDAP server using SSL) between the proxy and the LDAP server.
Admin distinguished name (DN)	The distinguished name of an account to which to bind, in order to perform searches. This "search" account is a read-only account used to do searches. The **admin** account can be used as the "search" account. If no admin DN is specified, then no bind is attempted. This parameter is only required when a site does not allow anonymous searches.
Admin password	A password for the "search" account created on the LDAP server.
LDAP Search Parameters	
Search type	The certificate-based authorization method that the proxy uses when searching the LDAP database (described in *Certificate-based authorization*, on page 7-44). Possible values are **USER**, **CERTMAP**, and **CERT**.
User base	The search base for the subtree used by the **USER** and **CERT** search types. A typical search base is: **ou=people,dc=company,dc=com**.
Certificate map base	The search base for the subtree used by the **CERTMAP** search type. A typical search base is similar to the following: **ou=people,dc=company,dc=com**.
User key	The name of the attribute in the LDAP database that specifies a user ID. Used by the **USER** search type. A typical example of a user key value is **uid**.
Certificate map key	The name of the certificate map found in the LDAP database. Used by the **CERTMAP** search type.

Table 7.6 *Configurable authorization model parameters*

Authorization model parameters	Description
Certificate serial number	The serial number of a certificate. Used for **CERTMAP** searches only, this parameter is used in combination with the certificate's issuer name, and replaces the subject DN. (A subject DN = country + state + city + org + common name + email.)
LDAP Filtering Parameters	
Group base	The search base for the subtree used by group searches. This parameter is only used when specifying valid groups. A typical search base would be similar to the following: **ou=people,dc=company,dc=com**.
Group key	The name of the attribute in the LDAP database that specifies the group name in the group subtree.
Group member key	The name of the attribute in the LDAP database that specifies members (DNs) of a group.
Valid groups	A list of groups to which a user must belong in order to be authorized.
Role key	The name of the attribute in the LDAP database that specifies a user's authorization roles. This key is used only when using valid roles (as described in the following entry).
Valid roles	A list of roles. A user must be assigned to one of these roles in order to be authorized. Valid roles in this list are delimited by a space.

Table 7.6 *Configurable authorization model parameters*

◆ Note

For an example of an authorization model, see Figure 7.10.

To create an authorization model using the Configuration utility

1. In the navigation pane, click **Proxies**.
   The Proxies screen opens.
2. Click the Authorization Model tab**.**
   This displays the SSL Proxy Authorization Model List screen. Any authorization models that you have defined are listed**.**
3. Click **Add.**
   This invokes the Add Authorization Model Wizard and displays the Configure Basic Properties screen.
4. Type a name for the authorization model and a value for the **Cache Timeout** parameter. Note that the **Method** parameter has only one allowable value, **LDAP**, and therefore cannot be changed.
5. Click **Next.**
   This displays the Configure LDAP General Properties screen.

6. Type values for the parameters listed and click **Next**.
   This displays the Configure LDAP Search Properties screen.
7. Enter a value for the **User Key** parameter.
8. Select either the **User**, **Certificate**, or **Certificate Map** button, and type in the corresponding value or values for that search type.
9. If you chose the **User search type**, click **Next**. Otherwise, proceed to Step 11.
   Clicking **Next** displays the Configure LDAP Filtering Properties screen.
10. Type in values for the **Group** and **Role** parameters listed and click **Done**.
11. If you chose the **Certificate** or **Certificate Map** search type, click **Done**.

To create an authorization model from the command line

Use the **bigpipe authz** command, as shown in the following command syntax.

```
b authz <name> [method (ldap|...)]
[cachetimeout <number>]
[ldap searchtype (user|certmap|cert)]
[ldap servers <server list>]
[ldap secure (enable | disable)]
[ldap admindn <string>]
[ldap adminpw <string>]
[ldap user base <string>]
[ldap user key <string>]
[ldap certmap base <string>]
[ldap certmap key <string>]
[ldap certmap useserial <string>]
[ldap group base <string>]
[ldap group key <string>]
[ldap group member key <string>]
[ldap valid groups <string list>]
[ldap role key <string>]
[ldap valid roles <string list>]
```

◆ Note

O*nce you have created an authorization model, you must associate that model with an SSL proxy. For detailed procedures,* see ***Associating an authorization model with an SSL proxy****, on page 7-50.*

Viewing authorization model properties

Through the Configuration utility, you can view at a glance all of the parameter values that make up an existing authorization model.

To view the properties of an authorization model using the Configuration utility

1. In the navigation pane, click **Proxies**.
   The Proxies screen opens.
2. Click the Authorization Model tab.
   This displays the SSL Proxy Authorization Model List screen, which lists any authorization models that you have defined.
3. Click an authorization model.
   This displays all of the parameter values defined for that model.

To view the properties of an authorization model from the command line

```
b proxy <ip_addr>:<service> authz models show
```

Configuring client authorization options on an SSL proxy

Table 7.7 lists and describes the authorization options that you can configure on an SSL proxy, after you have defined an authorization model.

SSL proxy authorization options	Description
Specifying an authorization header	Creates a standard HTTP header for authorization that is set to a server with the UID from the LDAP database.
Specifying a remote user header	Creates a standard HTTP header for remote users that is set to the UID used in the authorization.
Associating an authorization model with an SSL proxy	Specifies the name of an authorization model that you want to associate with an SSL proxy.
Setting the authorization **failure** mode	Determines, on authorization failure, whether the BIG-IP system terminates a client connection or allows the connection to pass through.
Specifying an authorization failure user name	Specifies a user name that is used when the user name in the LDAP database cannot be accessed due to authorization failure.

Table 7.7 *Configurable SSL proxy authorization options*

Specifying an authorization header

This option creates an authorization header. The value of this option can be either **true** or **false**. If **true**, the authorization header is created and set to the value of a server with the uid from the LDAP database. If **false**, no authorization header is created. The default value is **false**.

To insert an authorization header using the Configuration utility

1. From navigation pane, click **Proxies**.

2. Click the **Add** button.
3. In the **Insert Authorization Header** box, click the check box. This sets the value to **true**.
4. Click **Apply**.

To insert an authorization header from the command line

```
b proxy <ip_addr>:<service> authz set auth hdr enable
```

Specifying a remote user header

This option creates a remote user header. The value of this option can be either **true** or **false**. If **true**, the HTTP remote user header is set to the UID used in the authorization. If **false**, the HTTP remote user header is not created or modified. The default value is **false**.

To insert a remote user header using the Configuration utility

1. From navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Insert Remote-User Header** box, click the check box. This sets the value to **true**.
4. Click **Done**.

To specify a remote user header from the command line

```
b proxy <ip_addr>:<service> authz set remoteuser hdr \ enable
```

Associating an authorization model with an SSL proxy

This option specifies the name of an existing authorization model that you want to associate with an SSL proxy. You can make this association either at the time that you are creating the SSL proxy, or at any time after creating the proxy.

The following procedures show how to assoicate an authorization model with an existing SSL proxy.

To associate an authorization model with an existing SSL proxy using the Configuration utility

1. In the navigation pane, click **Proxies**.
   The Proxies screen opens.
2. Click a proxy name.
3. Click the Advanced Properties tab.
4. From the **Authorization Model** list, select the name of an authorization model that you previously created using the Add Authorization Model Wizard.

5. Click **Done**.

To associate an authorization model with an existing SSL proxy from the command line

Use the **bigpipe proxy** command, as follows:

```
b proxy <ip_addr>:<service> authz models <models list>
```

Setting the authorization failure mode

This option determines, on authorization failure, whether the BIG-IP system terminates a client connection or allows the connection to pass through. Allowed values are **accept** and **reject**. The default value is **reject**.

To set the authorization failure mode using the Configuration utility

1. From navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Authorization Failure Mode** box, select either **accept** or **reject**.
4. Click **Done**.

To set the authorization failure mode from the command line

```
b proxy <ip_addr>:<service> authz onfailure model accept | \
reject
```

Specifying an authorization failure user name

This option specifies a user name that is used when the user name in the LDAP database cannot be accessed due to authorization failure. This option only applies when the **Authorization Failure Mode** option is set to **accept**.

To specify an authorization failure user name using the Configuration utility

1. From navigation pane, click **Proxies**.
2. Click the **Add** button.
3. In the **Authorization Failure User Name** box, type a user name to be used in the header.
4. Click **Done**.

To specify an authorization failure user name from the command line

```
b proxy <ip_addr>:<service> authz onfailure username <username>
```

Network traffic control

The SSL proxy provides several configuration options that you can use to control your SSL traffic in certain ways. Table 7.8 summarizes these options.

Configuration Option	Description
Inserting headers into HTTP requests	Controls traffic by inserting custom headers, cipher specifications, client certificate fields, and session IDs into HTTP requests.
Rewriting HTTP redirections	Rewrites any HTTP redirection, as a way of ensuring that a connection remains on a secure channel.
Adding a lasthop pool to an SSL proxy	Sets up a lasthop pool that takes precedence over the **auto_lasthop** global variable. Replies are sent back through a router defined in the lasthop pool instead of the same router through which the request was originally received.
Disabling ARP requests	Disables all ARP requests for a proxy IP address.
Configuring proxy failover	Configures a redundant system so that failover from one unit to another occurs automatically, in the event of a cryptographic accelerator failure.

Table 7.8 *Options for configuring the SSL proxy to control network traffic*

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

If any of these header types is inserted into a valid HTTP request, the SSL proxy places the headers so that they immediately follow the first line of the request. If more than one header is inserted, the headers are inserted in the order listed above.

Security Considerations

Before using the header insertion feature, you should be aware of the following security considerations:

- If a client request uses a non-standard HTTP method, or the request is pipelined, the SSL proxy might not be able to insert headers, and in some cases, the entire connection might fail. Moreover, when the request does *not* fail, it is possible to mistake HTTP headers supplied by the client for those inserted by the SSL proxy. Therefore, if you are making security decisions based on the value of inserted headers, you should ensure that the client request uses a standard HTTP method or that the request is not

pipelined. The standard HTTP methods that the BIG-IP system currently supports are: **OPTIONS**, **GET**, **HEAD**, **POST**, **PUT**, **DELETE**, **TRACE**, **CONNECT**, as well as all currently-standardized WebDAV methods.

- Using the HTTP header insertion feature is not recommended when the SSL proxy targets a virtual server that is directly accessible to client connections. This is because a header could be mistakenly construed by the system as an indication that the connection came through the SSL proxy. Instead, the header could have been generated by the client itself when connecting directly to the virtual server. If you have configured your system to deliver certain content only to connections terminated by the SSL proxy, you should avoid using the proxy header insertion feature, when that proxy targets an externally-accessible virtual server. It is recommended that virtual servers intended to be accessed by SSL proxy connectinos only be configured with IP addresses on the loopback interface, for example **127.0.0.10**.

*The header insertion feature applies to HTTP requests only and should be enabled only on proxies that are configured to listen on HTTPS ports, such as port **443**.*

The following sections describe the four header types that you can insert into HTTP requests.

A custom HTTP header

When adding an SSL proxy, you can configure the proxy to insert a string of your choice, consisting of up to 253 characters, 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 \"header:value\"
```

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.

An inserted 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. For example, the following is a possible cipher specification inserted as a header into an HTTP request:

```
"SSLClientCipher: DES-CBC3-SHA, version=TLSv1/SSLv3, bits=168"
```

The ability to insert a cipher specification into a client request is useful because you can then build a rule that directs traffic based on the cipher specified in the request header. For example, you can build a rule, using the **http_header** rule variable,that directs requests with unacceptable cipher strengths to a *cipher upgrade* path, instead of discarding the sessions altogether.

Figure 7.11 shows an example of a rule that directs traffic based on the cipher specified in the request header.

```
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 7.11 *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 almost every field of a client certificate. This feature is most useful when:

- You have configured the SSL proxy to authenticate clients with the ***request*** option. 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 system or a server could then perform actions such as redirecting the request to another server, or sending different content back to the client. For more information, see *Configuring the presentation of client certificates*, on page 7-18.
- 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 7.12 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 7.12 *A rule based on certificate status specified in an HTTP header*

Table 7.9 shows the client certificate headers that the SSL proxy can insert into a client request. For each header, the format, description, and keyword are shown.

Header Name	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	SSLClientCertSignatureAlgorithm: [alg]	The signature algorithm of the certificate.
Issuer of the certificate	SSLClientCertIssuer: [issuer]	The issuer of the certificate.
Certificate validity dates	SSLClientCertNotValidBefore: [before] SSLClientCertNotValidAfter: [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	SSLClientCertSubject: [subject]	The subject of the certificate.
Public key of the subject	SSLClientCertSubjectPublicKey: [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	SSLClientCertHash: [hash]	The MD5 hash of the client certificate.

Table 7.9 *Formats of client certificate headers*

◆ Tip

When inserting a client certificate into a request, it is recommended for performance reasons that you insert the hash of a certificate, rather than the entire certificate itself. You can do this by using the ***SSLClientCertHash: [hash]*** *header.*

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 section, 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>
```

◆ Note

*The **status** header is always inserted automatically whenever one or more client certificate headers are configured to be inserted, and therefore is not specified in the Configuration utility or on the command line.*

Client session IDs

Client session IDs inserted as headers into HTTP requests, combined with use of universal persistence, provides a way to implement persistence for terminated SSL connections.

For unterminated connections, that is, when you have not created an SSL proxy, you can implement SSL persistence simply by configuring the SSL persistence attribute (**ssl**) on the pool that is receiving the SSL connections.

For terminated connections, that is, when you *have* created an SSL proxy, you implement SSL persistence in a different way--by inserting client session IDs as headers into HTTP requests, and then configuring universal persistence on the target pool. The SSL proxy contains a configuration option that you can use to insert a client session ID as a header into an HTTP request.

◆ Note

*The **ssl** persistence type is not supported on any virtual server that is the target of an SSL proxy. However, other persistence types are supported on vitual servers to which proxy connections are targeted. For information on persistence types, see Chapter 4, Pools.*

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.

Note

For information on enabling universal persistence based on client session IDs , see Chapter 4, Pools.

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 section, 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]
```

Rewriting HTTP redirection

One of the attributes you can set on a BIG-IP pool is HTTP redirection. Configured as a pool attribute, HTTP redirection redirects a client request to another protocol identifier, host name, port number, or URI path. For example, a pool might be configured to redirect a request from the HTTPS protocol to the HTTP protocol.

When a client request is redirected from the HTTPS to the HTTP protocol, an SSL proxy can rewrite that redirection back 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 reason to configure a proxy to rewrite HTTP redirections pertains to IIS and Netscape web-server environments. Without a BIG-IP proxy, a web server running IIS and Netscape redirects a request incorrectly, if the original request included a malformed directory name (without a trailing slash [/]). An SSL proxy has the ability to rewrite this incorrect redirection.

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.

◆ Note

An alternative to configuring an SSL proxy to rewrite this type of redirection is to install a special BIG-IP file, ***redirectfilter.dll****, on your IIS server. Once this file is installed, the IIS server can handle any rewriting of HTTP redirections, without the help of the SSL proxy. For background information, see* ***Redirecting HTTP requests****, on page 4-48.*

To enable the proxy to rewrite HTTP redirections, you simply specify, through either the Configuration utility or the **bigpipe proxy** command, the way that you want the proxy to handle URIs during the rewrite. Once enabled, this feature rewrites the protocol name and the port number. The following sections describe proxy behavior when the rewrite feature is enabled.

Selecting URIs to rewrite

When configuring the SSL proxy to rewrite HTTP redirections, you specify whether the proxy should rewrite only those URIs **matching** the URI originally requested by the client (minus an optional 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 7.10 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 7.10 *Examples of rewriting HTTP redirections with SSL proxy listening on port* ***443***

Table 7.11 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 7.11 *Examples of rewriting HTTP redirections with SSL proxy listening on port* ***4443***

Rewriting the protocol name

When configured to rewrite HTTP redirections, the SSL proxy rewrites 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 rewriting the protocol name from HTTP to HTTPS, the SSL proxy rewrites 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**.

Configuration procedures for rewriting HTTP redirections

You can configure the rewrite feature using either the Configuration utility or the **bigpipe proxy** command.

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 section, 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>
```

Adding a last hop pool to an SSL proxy

In cases where you have more than one router sending connections to a BIG-IP system, 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 takes 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
```

Disabling ARP resquests

By default, the BIG-IP system responds to ARP requests for proxy addresses 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**.

Configuring SSL proxy failover

If you have a BIG-IP system redundant configuration, you might be able to 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 system, after having had an initial success communicating with the cryptographic accelerator module, subsequently receives a hardware error.

Not all cryptographic accelerator modules generate hardware errors upon failure. Thus, the ability to configure automatic SSL proxy failover depends on the type of accelerator module that you are using.

This option is configured globally, and by default is set to **disable**.

Important

*For redundant system configurations, you can also configure an option known as connection mirroring, on an individual virtual server . **Connection mirroring** mirrors connection information to the peer unit during failover. Be aware, however, that you cannot enable connection mirroring on a virtual server that is the target of an SSL proxy.*

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 **Enabled** box of the **SSL Accelerator Failure** area, check the 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>
```

Other SSL protocol options

In addition to the configuration options on an SSL proxy that allow you to control SSL-related functions such as certificate validation, verification, and revocation, the proxy offers other options related to the SSL protocol. Figure 7.12 lists and describes these options

Configuration Option	Description
Configuring invalid SSL protocol versions	Specifies the SSL protocol versions that the proxy will consider to be invalid.
Configuring the SSL session cache	Configures both the size and the timeout values for the SSL session cache.
Configuring SSL shutdowns	Configures the proxy to force clean SSL shutdowns, and to allow SSL sessions to resume after unclean shutdowns.

Table 7.12 *Other SSL configuration options*

The remainder of this chapter describes these options and how to configure them.

Configuring invalid protocol versions

When configuring protocol versions, you must ensure that the protocol versions configured for the SSL proxy match those of the proxy's peer. That is, protocol versions for the client-side SSL proxy must match those of the client, and protocol versions for the server-side SSL proxy must match those of the server. Thus, for both client-side and server-side SSL connections, you can specify the protocol versions that are not 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.

◆ Note

At a minimum, it is recommended that you specify SSLv2 as invalid.

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 section or the Server-side Connections Do Not Use These SSL Versions section, check the appropriate boxes.
4. Click **Done**.

To specify invalid SSL protocol versions from the command line

To specify invalid SSL protocol versions, use the following syntax:

```
b proxy <ip>:<service> [clientssl] invalid [SSLv2] [SSLv3] [TLSv1]
b proxy <ip>:<service> serverssl invalid [SSLv2] [SSLv3] [TLSv1]
```

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, you can configure the client-side values on a per-proxy basis. For server-side SSL connections, however, the proxy maintains a single session cache. Thus, you must configure server-side session cache values 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.You can set client-side timeout values 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 significantly 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 **0**, 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 using the Configuration utility

1. In the navigation pane, click **System**.
2. Click 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 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. A value of **0** disallows session caching.

You can configure the client-side values for the maximum size of the session cache on a per-proxy basis. You can configure a single, server-side value for the maximum size of the session cache 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 shutdowns

With respect to the shutdown of SSL connections, you can configure two global options on the BIG-IP system:

- 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 **Strict SSL Shutdown Compliance (causes slower IE sessions)** box, check or clear the check box.
   Clearing the box prevents the proxy from resuming SSL sessions after an unclean shutdown. Checking the box enables the proxy to resume SSL sessions after an unclean shutdown.
4. Click **Done**.

To configure the SSL proxy to resume SSL sessions from the command line

To prevent the SSL proxy from resuming SSL sessions after an unclean shutdown, you can enable the **strict resume** global attribute on the SSL proxy, as follows:

```
b global sslproxy strict resume enable
```

To allow the SSL proxy to resume SSL sessions after an unclean shutdown, you can disable the **strict resume** global attribute on the SSL proxy, as follows:

```
b global sslproxy strict resume diable
```

8

Nodes

- Introducing nodes
- Configuration options

Introducing nodes

Nodes are the network devices to which the BIG-IP system 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, using either the Configuration utility or the **bigpipe node** command.

Configuration options

The attributes you can configure for a node are listed in Table 8.1.

Node Attributes	Description
Enable/Disable nodes	You can enable or disable nodes independent of a load balancing pool.
Mark nodes as up or down	You can set a node to **up** or **down**.
Setting Connection limits	You can place a connection limit on a node.
Associate a node with a health monitor	You can associate a health monitor with a node, creating an instance of that monitor.
Displaying node status	You can display the status of a node, as well as a summary of connection statistics.
Resetting node statistics	You can reset the statistics for an individual node address.
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 modes and persistence types defined in the pool to control connections handled by the node.

Table 8.1 *The attributes you can configure for a node.*

The following sections describe the procedures for configuring these options.

Enabling and disabling 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 and disable nodes and node addresses

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

```
b node 192.168.21.1 enable
```

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

```
b node 192.168.21.1 disable
```

Marking nodes and node ports as 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 nodes and node ports up or down using the Configuration utility

1. In the navigation pane, click **Nodes**.
   The Nodes screen opens.
2. In the list of nodes, click a node address.
   The properties page for that node opens.
3. To mark a node as **down**, clear the **Enable Connections** check box.
   To mark a node as **up**, click the **Enable Connections** check box.
4. Click **Apply**.

To mark nodes and node ports up or down from the command line

To mark a node as **down**, specify the **bigpipe 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.) For example:

```
b node 192.168.21.1 dow
```

To mark a node as **up**, use the **bigpipe node** command with the **up** option. For example:

```
b node 192.168.21.1 up
```

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

```
b node 192.168.21.1:80 down
```

To mark a particular service as **up**, use the **bigpipe node** command with **up** option. For example:

```
b node 192.168.21.1:80 up
```

Note

*If you mark a node as **down**, you cannot use the **bigpipe load** command to undo the **down** state.*

Setting connection limits for nodes

Using the Configuration utility or the **bigpipe node** command, you can limit the number of concurrent connections that a node will allow.

To set connection limits for nodes using the Configuration utility

1. In the navigation pane, click **Nodes**.
   The Nodes screen opens.
2. In the list of nodes, click a node address.
   The properties page for that node opens.
3. In the **Connection Limit** box, type a number representing the maximum concurrent connections allowed on a node.
4. Click **Apply**.

To set connection limits for nodes from the command line

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).

Associating monitors with nodes

To implement a health or performance monitor, you must associate that monitor with a node.

To associate a monitor with a node

Use the following command line syntax to associate a monitor with a node:

```
b node <node> monitor use <monitor>
```

A monitor can be placed on multiple nodes, and a node can have multiple monitors placed on it.

To associate a monitor with multiple nodes:

```
b node <node_list> monitor use <monitor>
```

To associate multiple monitors with a node:

```
b node <node> monitor use <monitor1> and <monitor2>...
```

For more information on using the **bigpipe node** command with monitors, refer to Chapter 11, *Monitors*.

Displaying node status

When you issue the **bigpipe node show** command, the BIG-IP system 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 service.

To display status of all nodes using the Configuration utility

In the navigation pane, click **Nodes**. The Nodes screen opens and displays the status of all nodes.

To display status of all nodes from the command line

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 8.1 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 8.1 *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
```

Resetting node statistics

You can reset statistics for an individual node address. To do this, use the following command line syntax:

```
b node [<node_ip>:<service>] stats reset
```

Adding nodes to pools

You can add a node as a member of a load balancing pool. For more information, see Chapter 4, *Pools*.

9

Services

- Introducing services
- Configuration options

Introducing services

The BIG-IP system supports a variety of ***services*** that are standard Internet applications, 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 to 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 system. Because this command is global, you need to open access to a service only once; you do not need to open access to a service for each instance of a virtual server that uses it.

Configuration options

You can set a number of attributes on a service, such as allowing access to a service and setting the number of conncurrent connections that a service will allow. You can set these services using the **bigpipe service** command.

◆ Note

Any time you create a virtual server and define a service with the Configuration utility or the bigpipe command, network traffic is automatically enabled for that service.

Table 9.1 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 system. In order for the BIG-IP system to load balance traffic, you must enable access to the service on which the BIG-IP system will receive traffic.
Set connection limits	You can define a connection limit for a service so that a flood of connections does not overload the BIG-IP system.
Enable and disable TCP and UDP	You can enable or disable TCP and UDP for specific services.
Set idle connection timeouts	You can set the idle connection timeout to close idle connections.
Display service settings	You can issue a command to display the settings for any of the above options that you have configured.

Table 9.1 *Service attributes*

Allowing access to services

Setting this attribute allows the specified service to accept network connections.

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
```

Setting connection limits on services

You can set the maximum number of concurrent connections allowed on a service. Note that you can configure this setting for one or more services.

To set connection limits on services

To set a limit on the number of connections that a node will accept, use the following command line syntax:

```
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
```

Enabling and disabling TCP and UDP for services

You can enable or disable TCP or UDP for specific services.

To enable or disable TCP for services

The default setting for all TCP 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

The default setting for all UDP 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
```

Setting the idle connection timeout

The idle connection timeout attribute specifies, for TCP or UDP services, the number of seconds that transpires before an idle connection is dropped.

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
```

Displaying service settings

You can display the settings that you specified for the available service attributes. Thus, this display shows the maximum number of concurrent connections that each service allows, whether TCP or UDP is enabled on each service, and their idle connection timeout values.

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 or services:

```
b service <service> [...<service>] show
```

For example, the command **b service http show** displays the output shown in Figure 9.1.

```
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 9.1 *Sample output of the* ***bigpipe service show*** *command*

10

Address Translation: SNATs, NATs, and IP Forwarding

- Introducing address translation
- SNATs
- NATs
- IP forwarding

Introducing address translation

The BIG-IP system 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, see Chapter 6, *Virtual Servers*. 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 10.1.

Attributes	Description
Global SNAT properties	Before you configure a SNAT, you can configure global properties for all SNATs on the BIG-IP system. 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 a BIG-IP system to automatically map a translation address.

Table 10.1 *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 10-9.

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 system 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 system 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 10-9.*

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 the BIG-IP system 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 the BIG-IP system 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 syntax 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

The BIG-IP system includes a feature called SNAT automapping. When you map a SNAT automatically, rather than manually, you enable the BIG-IP system to choose the translation IP address. You also enable the BIG-IP system 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 system, 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 system 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 system 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 **NAT**s 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 10-9.*

Adding automapped SNATs for standard (active-standby) configurations

When enabling SNAT automapping for VLANs, the BIG-IP system 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 system 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 system 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 10-9.

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 NATs screen displays.
2. Click the SNATs tab.
3. Click the **Add** button.
   The Add SNAT screen opens.
4. In the **Translation Address** 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 10-7.

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 system 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 10-7.*

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. 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

The BIG-IP system 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 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.

Disabling ARP requests

By default, the BIG-IP system 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**.

Configuring a cache server

To ensure that a cache server or remote origin server responds to the BIG-IP system rather than to the original cache server that generated the missed request, the BIG-IP system 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**
  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 10-2.
- **Create a SNAT auto-mapping 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 system and not directly to the client. (If the BIG-IP system replied directly to the client, the next request would then go directly to the origin server, removing the BIG-IP system from the loop.) For more information about SNAT automapping, see *Configuring SNAT automapping*, on page 10-6.

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 10.2 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 system. 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 system is configured to run in active-active mode.

Table 10.2 *The attributes you can configure for a NAT*

The IP addresses that identify nodes on the BIG-IP system 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 system, rather than having the BIG-IP system 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 system 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.
4. Click **Done**.

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 system 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 10.1).

```
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 10.1 *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 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.

Disabling ARP requests

By default, the BIG-IP system 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 system.
- 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 Chapter 4, Pools and Chapter 6, Virtual Servers.

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 enable IP forwarding, the BIG-IP system 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 10.3 shows options associated with IP forwarding.

Option	Description
Enable IP forwarding globally	You can enable IP forwarding globally for the BIG-IP system, 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 system.
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 Chapter 4, *Pools*.
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 Chapter 6, *Virtual Servers*.

Table 10.3 *The attributes you can configure for IP forwarding*

The following sections describe the procedures for configuring these options.

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 system 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 system, 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 system. 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 system or by an individual virtual server. For more information, see Chapter 4, *Pools*.

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 system or by a specific pool. For more information, see Chapter 6, *Virtual Servers*.

11

Monitors

- Introducing monitors
- Choosing a monitor
- Configuring a monitor
- Associating monitors with nodes
- Showing, disabling, and deleting monitors

Introducing monitors

An important feature that the BIG-IP system provides is a load-balancing tool called monitors. ***Monitors*** verify connections and services on nodes that are members of load balancing pools. A monitor can be either a health monitor or a performance monitor, designed to check the status of a node or service on an ongoing basis, at a set interval. If the node or service being checked does not respond within a specified timeout period, or the status of the node indicates that the performance of the node is degraded, the BIG-IP system can redirect the traffic to another node.

- **Monitor types**
  The BIG-IP system includes several types of monitors from which to choose, depending on the type of node or service to be monitored. For example, an **http** monitor allows you to monitor the availability of the HTTP service on a node. A **wmi** monitor allows you to monitor the performance of a node that is running the Windows Management Instrumentation (WMI) software. An **icmp** monitor simply determines whether the status of the node address itself is **up** or **down**. For more information on monitor types, see *Summary of monitor types*, on page 11-2 and *Choosing a monitor*, on page 11-7.

- **Monitor attributes**
  Each monitor type contains a set of attributes to which default values are assigned. For example, the **icmp** monitor has the following attributes and values:

```
monitor icmp {
    #type icmp
    interval 5
    timeout 16
    dest *
    }
```

  The attributes above specify that the **icmp** monitor is configured to check the status of an IP address every five seconds, and to time out every 16 seconds. The destination IP address that the monitor checks is specified by the **dest** attribute. In the above example, all IP addresses with which the monitor is associated are checked. For more information on monitor attributes, see *Summary of monitor attributes*, on page 11-4 and *Choosing a monitor*, on page 11-7.

- **Monitor implementation**
  To implement a monitor, you simply open the Confguration utility, display the **Monitors** screen, and select a monitor for the type of node or service you want to check. If you want to adjust the default values of the monitor attributes, you can do that, thus creating a monitor that is customized for your particular needs. You then associate the monitor with the appropriate node. For more information on implementing a monitor, see *Configuring a monitor*, on page 11-20.

Summary of monitor types

The BIG-IP system includes many different types of monitors, each designed to perform a specific type of monitoring. Table 11.1 lists the BIG-IP monitors that you can configure for controlling your network traffic.

Monitors	Description
Simple monitors	
icmp	Checks the status of a node, using Internet Control Message Protocol (ICMP).
tcp_echo	Checks the status of a node, using Transmission Control Protocol (TCP).
Extended Content Verification (ECV) monitors	
tcp	Verifies the Transmission Control Protocol (TCP) service by attempting to receive specific content from a node.
http	Verifies the Hypertext Transfer Protocol (HTTP) service by attempting to receive specific content from a web page.
https	Verifies the Hypertext Transfer Protocol Secure (HTTPS) service by attempting to receive specific content from a web page protected by Secure Socket Layer (SSL) security.
https_443	Verifies the Hypertext Transfer Protocol Secure (HTTPS) service by attempting to receive specific content from a web page protected by Secure Socket Layer (SSL) security.
Extended Application Verification (EAV) monitors	
external	Allows users to monitor services using their own programs.
ftp	Verifies the File Transfer Protocol (ftp) service by attempting to download a specific file to the **/var/tmp** directory on the BIG-IP system.
pop3	Verifies the Post Office Protocol (pop3) service by attempting to connect to a node, log on as the specified user, and log off.
smtp	Checks the status of a node by issuing standard Simple Mail Transport Protocol (SMTP) commands.
nntp	Verifies the Usenet News protocol (NNTP) service by attempting to retrieve a newsgroup identification string from the server.
sql	Verifies SQL-based services by attempting to perform an SQL login to a service.
imap	Verifies the Internet Message Access Protocol (IMAP) by attempting to retrieve a specified message number. This monitor is similar to the **pop3** monitor.
radius	Verifies the Remote Access Dial-in User Service (RADIUS) service by attempting to authenticate the specified user.
ldap	Verifies the Lightweight Directory Access Protocol (LDAP) service by attempting to authenticate the specified user.

Table 11.1 *Monitor types available on a BIG-IP system*

Monitors	Description
udp	Verifies the User Datagram Protocol (UDP) service by attempting to send UDP packets to a node and receiving a reply.
real_server	Checks the performance of a node that is running the RealServer data collection agent and then dynamically load balances traffic accordingly.
wmi	Checks the performance of a node that is running the Windows Management Infrastructure (WMI) data collection agent and then dynamically load balances traffic accordingly.
snmp_dca	Checks the current CPU, memory, and disk usage of a node that is running an SNMP data collection agent and then dynamically load balances traffic accordingly.
snmp_dca_base	Checks the current user usage of a node that is running an SNMP data collection agent and then dynamically load balances traffic accordingly.

Table 11.1 *Monitor types available on a BIG-IP system*

Summary of monitor attributes

Table 11.2 contains a complete list of all possible monitor attributes, along with their definitions. Note that each monitor template contains only a subset of these attributes.

Attribute	Definition
interval <seconds>	Ping frequency time interval in seconds.
timeout <seconds>	Ping timeout in seconds.
dest <node_addr>	Ping destination node. Usually has a value of ***** for node address monitors and ***:*** 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 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.

Table 11.2 *List of all possible monitor attributes*

Attribute	Definition
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.
sendpackets <num>	Number of packets to send when using the **udp** monitor.
timeoutpackets <seconds>	Timeout in seconds for receiving UDP packets.

Table 11.2 *List of all possible monitor attributes*

Working with monitor templates

The BIG-IP system provides a variety of monitors in template form. A ***monitor template*** consists of a set of attributes, with default values assigned to them. Some of these monitors are usable as is (assuming that their default values are acceptable). These are called ***default monitors***. In most cases, however, a monitor template is used strictly as a basis for creating a new monitor that changes the attribute values. These monitors are called ***custom monitors***.

Once you have chosen or created the monitor you want to use to monitor a node, you use the Configuration utility or the **bigpipe node** command to associate the monitor with that node.

All monitor templates are contained in one of the following read-only files, depending on your licensed product:

- **/etc/base_monitors.3dns**
- **/etc/base_monitors.flb**
- **/etc/base_monitors.lb**
- **/etc/base_monitors.clb**
- **/etc/base_monitors.ha**
- **/etc/base_monitors.ssl**
- **/etc/base_monitors.lc**

Default monitors

A ***default monitor*** is a monitor template that is used as is, that is, with the default values assigned to its attributes. An example of a default monitor is the **icmp** monitor, which uses the default values contained in the **icmp** monitor template. A health monitor of the simplest type, the **icmp** monitor is automatically associated with every node in a pool and checks a node address through a **ping** response. This automatic association is in the form of the following entry in the bigip.conf file:

```
node * monitor use icmp
```

Figure 11.1 shows the **icmp** monitor template.

```
monitor type icmp {
    interval 5
    timeout 16
    dest *
}
```

Figure 11.1 *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.)

To change the interval and timeout values of a default monitor, you need to configure a custom monitor, as described in the following section.

Custom monitors

A ***custom monitor*** inherits the attributes and the default values of a monitor template. Once you have created the custom monitor, you can change the attribute values as required.

For example, you can create a custom monitor called **my_icmp**, based on the **icmp** monitor template. When creating the **my_icmp** custom monitor, you can specify those attribute values that you want to change. For example, if you want to change the timeout value from the default value of **16** to a value of **20**, you can define the custom monitor as follows:

```
b monitor my_icmp '{ use icmp timeout 20 }'
```

This creates a new monitor called **my_icmp**, based on the monitor template **icmp**, with a timeout value of **20**. As shown in Figure 11.2, you can display the custom monitor using the command **b monitor my_icmp show**.

```
monitor my_icmp{
    #type icmp
    "icmp"
    interval 5
    timeout 20
    dest *
    }
```

Figure 11.2 *Custom* ***icmp*** *monitor*

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 choose 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**.

For procedures on selecting and configuring a monitor template, see *Configuring a monitor*, on page 11-20.

Choosing a monitor

The first task in implementing a BIG-IP monitor is to select a monitor template. Monitor templates fall into three categories:

- **Simple monitors**
  These are health monitors that monitor the status of a node.
- **Extended Content Verification (ECV) monitors**
  These are either health or performance monitors that verify service status by retrieving specific content from nodes.
- **External Application Verification (EAV) monitors**
  These are health monitors that verify service status by executing remote applications, using an external service-checker program.

The following sections list and describe these monitor types.

Simple monitors

Simple monitors are those that check only node addresses, and verify only simple connections. 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.*

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. The **icmp** template has no attributes other than the standard **interval**, **timeout**, and **dest**.

```
monitor icmp {
    #type icmp
    interval 5
    timeout 16
    dest *
    }
```

Figure 11.3 *The **icmp** monitor template*

The **transparent** mode is an option for monitors derived from the **icmp** template. For more information about **transparent** mode, refer to *Using transparent and reverse modes*, on page 11-23.

tcp_echo

The **tcp_echo** template verifies Transmission Control Protocol (TCP) connections. The check is successful if the BIG-IP system receives a response to a TCP ECHO message. The **tcp_echo** template 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, see *Using transparent and reverse modes*, on page 11-23.)

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 11.4 *The **tcp_echo** monitor template*

The **transparent** mode is an option for monitors derived from the **tcp_echo** template. For more information about **transparent** mode, refer to *Using transparent and reverse modes*, on page 11-23.

Extended Content Verification (ECV) monitors

ECV monitors use **send** and **recv** statements in an attempt to retrieve explicit content from nodes. These monitors include **http**, **https** and **tcp**.

Note

*The templates **http**, **https, https_443**, 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.*

tcp

A **tcp** monitor attempts to receive specific content sent over TCP. 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 11-23.

```
monitor tcp  {
    #type tcp
    interval 5
    timeout 16
    dest *:*
    send ""
    recv ""
    //
    //transparent
    }
```

***Figure 11.5** The **tcp** monitor template*

http

The **http** template is used to monitor Hypertext Transfer Protocol (HTTP) traffic. 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, a **username**, a **password**, and optional **get**, **url**, and **transparent** 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 to which the URL resolves. Both transparent and reverse modes are also options. For more information about transparent and reverse modes, refer to *Using transparent and reverse modes*, on page 11-23. For more information about the **get** and **url** statements, refer to *Using send, receive, url, and get statements*, on page 11-22. Figure 11.6 shows the http monitor template.

```
monitor http {
    #type http
    interval 5
    timeout 16
    dest *:*
    send "GET /"
    recv ""
    username ""
    password ""
    //get
    //url
    //
    //transparent
    //reverse
    }
```

Figure 11.6 *The* ***http*** *monitor template*

https and https_443

The **https** and **https_443** templates are used to monitor Hypertext Transfer Protocol Secure (HTTPS) traffic. An **https** or **https_443** 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** or **https_443** 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 to which the URL resolves. Figure 11.7 shows the **https** monitor template.

```
monitor https {
    #type https
    interval 5
    timeout 16
    dest *:*
    send "GET /"
    recv ""
    //get
    //url
    username ""
    password ""
    //reverse
    }
```

***Figure 11.7** The **https** monitor template*

Figure 11.8 shows the **https_443** monitor template.

```
monitor https_443 {
    #type https
    interval 5
    timeout 16
    dest *:443*
    send "GET /"
    recv ""
    //get
    //url
    username ""
    password ""
   //reverse
    }
```

***Figure 11.8** The **https_443** monitor template*

The **reverse** mode is an option for monitors derived from the **https** and **https_443** templates. For more information about **reverse** mode, refer to *Using transparent and reverse modes*, on page 11-23.

External Application Verification (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**, **sql**, **nntp**, **imap**, **ldap**, and **radius**. Also included is the template **external**, which has a **run** attribute to specify a user-added external monitor.

external

The BIG-IP system allows you to create your own monitor types**.** The **external** template is then used to find these user-supplied monitors and execute them.

The **external** monitor template includes a **run** attribute that you use to specify the name of your user-supplied monitor. By default, the **external** monitor searches the directory **/user/local/lib/pingers** for that monitor name. If the user-supplied monitor resides elsewhere, a fully qualified path name must be entered.

The **run** attribute of an **external** monitor requires the executable name of the user-supplied monitor, and the **args** attribute allows you to specify any command line arguments required.

Figure 11.9 shows the **external** monitor template, with the default attribute values.

```
monitor external {
    #type external
    interval 5
    timeout 16
    dest *:*
    run ""
    args ""
    }
```

Figure 11.9 *The* ***external*** *monitor template*

For example, suppose the program **my_pinger** is to be run with a **-q** option, so that it is 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 11.10.

```
monitor custom {
    use external
    run "/var/my_pinger"
    DIRECTORY "www/test_files"
    FILE "first_test" }
```

Figure 11.10 *A monitor derived from the external monitor template*

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.

ftp

The **ftp** template is used to monitor File Transfer Protocol (FTP) traffic. The monitor attempts to download a specified file to the **/var/tmp** directory, and if the file is retrieved, the check is successful. The **ftp** monitor takes a **get** statement, a **username**, and a **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 11-22.)

Figure 11.11 shows the **ftp** monitor template, with the default attribute values.

```
monitor ftp {
    #type ftp
    interval 10
    timeout 31
    dest *:*
    username ""
    password ""
    get ""
    //url
    }
```

Figure 11.11 *The **ftp** monitor template*

imap

The **imap** template is used to monitor Internet Message Access Protocol (IMAP) traffic. 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 11.12 *The **imap** monitor template*

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.*

ldap

The **ldap** template is used to monitor Lightweight Directory Access Protocol (LDAP) servers. LDAP implements standard X.500 for email 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 the search item.

```
monitor ldap {
    #type ldap
    interval 5
    timeout 16
    dest *:*
    username ""
    password ""
    base ""
    filter ""
    }
```

Figure 11.13 *The ldap monitor template*

nntp

The **nntp** template is used to monitor Usenet News traffic. 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, a **username** and **password**.

Figure 11.14 shows the **nntp** monitor attributes and their default values.

```
monitor nntp {
    #type nntp
    interval 5
    timeout 16
    dest *:*
    username ""
    password ""
    newsgroup ""
    }
```

Figure 11.14 *The **nntp** monitor template*

pop3

The **pop3** template is used to monitor Post Office Protocol (POP) traffic. 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 11.15 *The **pop3** monitor template*

radius

The **radius** template is used to monitor Remote Access Dial-in User Service (RADIUS) 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 11.16 *The **radius** monitor template*

real_server

The **real_server** performance monitor checks the performance of a node that is running the RealSystem Server data collection agent and then dynamically load balances traffic accordingly. Performance monitors are generally used with dynamic ratio load balancing. For more information on performance monitors and dynamic ratio load balancing, see Chapter 4, *Pools*.

◆ Note

Unlike health monitors, performance monitors do not report on the status of a node.

smtp

The **smtp** template is used to monitor Simple Mail Transport Protocol (SMTP) 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 11.17 *The* ***smtp*** *monitor template*

snmp_dca

The **snmp_dca** performance monitor 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 11.18 shows the **snmp_dca** monitor template, with the default attribute values.

```
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 11.18 ***snmp_dca*** *monitor template*

Performance monitors are generally used with dynamic ratio load balancing. For more information on performance monitors and dynamic ratio load balancing, see Chapter 4, *Pools*.

◆ Note

Unlike health monitors, performance monitors do not report on the status of a node.

snmp_dca_base

Like the **snmp_dca** monitor, the **snmp_dca_base** performance monitor 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 11.19 shows the **snmp_dca_base** monitor template, with the default attribute values.

```
monitor type snmp_dca_base {
    #type snmp_dca_base
    interval 10
    timeout 30
    dest *:161
    }
```

***Figure 11.19 snmp_dca_base** monitor template*

Performance monitors are generally used with dynamic ratio load balancing. For more information on performance monitors and dynamic ratio load balancing, see Chapter 4, *Pools*.

◆ Note

Unlike health monitors, performance monitors do not report on the status of a node.

sql

The **sql** template is used 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 11.20 *The **sql** monitor template*

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 system.

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.

udp

The **udp** template is used when sending User Datagram Protocol (UDP) packets. Designed to check the status of a UDP service, the **udp** monitor sends one or more UDP packets to a target node. Figure 11.21 shows the attributes of a **udp** monitor template, with the default values. As shown in this figure, the value in seconds of the **timeoutpackets** attribute should be lower than the value of the **interval** attribute.

```
monitor udp  {
    #type udp
    interval 5
    timeout 16
    dest *:*
    send ""
    sendpackets "2"
    timeoutpackets "3"
    }
```

Figure 11.21 *The **udp** monitor template*

When using a **udp** monitor to monitor a node address, you must also enable another node address monitor, such as **icmp**, to monitor the node address.

If the udp monitor reports node status as	And the node address monitor reports node address status as	Then the UDP service is...
up	up	up
up	down	down
down	up	down
down	down	down

Table 11.3 *Determining status of the UDP service*

Until both the **udp** monitor and the other node address monitor report the status of the UDP service as up, the UDP service receives no traffic.

wmi

The **wmi** performance monitor checks the performance of a node that is running the Windows Management Infrastructure (WMI) data collection agent and then dynamically load balances traffic accordingly.

Performance monitors are generally used with dynamic ratio load balancing. For more information on performance monitors and dynamic ratio load balancing, see Chapter 4, *Pools*.

◆ Note

Unlike health monitors, performance monitors do not report on the status of a node.

Configuring a monitor

The second task 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 any attributes for which default values need to be changed. You can also add any optional attributes, such as **transparent or reverse**, that are not present by default.

For special considerations related to changing the default values of monitor attributes, see *Changing attribute values*, on page 11-21.

Configuration procedures

You can configure a monitor from a template using the Configuration utility or 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 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}'
```

Changing attribute values

Each monitor has a set of attributes with default values assigned. The following sections contain information that is useful when changing these default values.

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, the value for the **dest** attribute in all monitor templates is set to the wildcard **"*"** or **"*:*"**. This value causes the monitor instance created for a node to take that node's address or address and port as its destination. You can, however, replace either or both wildcard symbols with an explicit **dest** value, by creating a custom monitor. This is referred to as ***node*** and ***port aliasing***. An explicit value for the **dest** attribute is used to force the instance destination to a specific address and/or port which may not be that of the node.

Specifying an explicit value for the **dest** attribute causes the monitor to ping that forced destination by an unspecified path. Suppose, for example, that the association that was performed using **http** instead used the custom monitor **my_http** with a **dest** value of ***:443**. The node association command is 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 11.22.

```
+- 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 11.22 *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 be 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 **11.12.11.22:80** it produces the following instances, (which are in fact one instance associated with three different nodes) as shown in Figure 11.23.

```
+- 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 11.23 *Node addresses aliased*

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 retrieves a default HTML page for a web site. To retrieve a specific page from a web site, 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 is 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. For the **ftp** monitor, the get attribute specifies the full path to the file to retrieve.

The attribute **url** takes the URL as a value and automatically fills in the **dest** value with the address to which the URL resolves. 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 normal and default behavior for a monitor is to ping the **dest** node by an unspecified route and to mark the node **up** if the test is successful. However, two other modes, transparent and reverse, are also available. These modes are optional and are **s**pecified by the **transparent** and **reverse** attributes.

Figure 11.4 shows the monitors that have the **transparent** and **reverse** attributes.

Monitor	Attribute	
tcp	transparent	reverse
http	transparent	reverse
https		reverse
tcp_echo	transparent	
icmp	transparent	

Table 11.4 *Monitors that have the transparent or reverse attributes*

- **Transparent mode**
  Sometimes it is necessary to ping the aliased destination through a transparent node. In ***transparent*** mode, the monitor is forced to ping through the node with which it is associated (usually a firewall) to the

dest node. (In other words, if there are two firewalls in a load balancing pool, the destination node is always pinged through the node specified and not through the node selected by the load balancing method.) In this way, the transparent node is tested--if there is no response, the transparent node is marked as **down**.

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 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 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**. For more information on associating monitors with nodes, see *Associating monitors with nodes*, on page 11-24.

- **Reverse mode**
  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 means that the web server was down.

Associating monitors with nodes

Now that you have created a monitor, the third and final task is to associate the monitor with the nodes to be monitored. This creates an instance of the monitor for each node. Associating a monitor with a node can be done either using the Configuration utility or the **bigpipe node** command.

While the term node association is applied generally with respect to associating monitors with nodes, there are actually three types of associations, based on whether the monitor is associated with an IP address and service, an IP address only, or a service only. These types are node association, node address association, and service association.

- ***Node association*** is the association of a monitor with an IP address and a service.
- ***Node address association*** is the association of a monitor with an IP address only.
- ***Service association*** is the association of a monitor with a service only. For a service association, a wildcard character (*) is used to represent all IP addresses.

To illustrate, the following is an example of a node association, where the monitor **http** is being associated with nodes **11.12.11.20:80, 11.12.11.21:80,** and **11.12.11.22:80**.

```
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 following command:

```
b node monitor show
```

The above command produces the output shown in Figure 11.24.

```
+- 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 11.24 *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 11.24.

Other ways to associate monitors with nodes are through wildcard specification, and through logical grouping.

Specifying wildcards

You can also use the wildcard character * to specify node addresses. You can use a wildcard node address association to create instances of the monitor for all node addresses load balanced by the BIG-IP system. For example:

```
b node * monitor use my_tcp_echo
```

A wildcard with a service association creates instances of the monitor for every node address configured to service that port:

```
b node *:80 monitor use my_http:
```

For detailed procedures on associating monitors with nodes, see *Configuration procedures*, on page 11-26.

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 are set to ***:***, then both monitor instances try to ping the default port **80**. This both defeats the purpose of the HTTPS monitor, and causes a configuration error, since two monitors are trying to ping the same address and port simultaneously.

Instead,you should give monitor **my_http** a **dest** value of ***:*** and give monitor **my_https** 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 11.25.

```
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 11.25 *Use of a monitor rule*

Configuration procedures

Using the Configuration utility or the **bigpipe node** command, you can create, show, or delete an association between a monitor and a node, node address, or service.

To associate a monitor using the Configuration utility

1. In the navigation pane, click **Monitors**.
   The Monitors screen opens.
2. Click the tab corresponding to the type of association you want:
   - 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 service only (that is, the service minus the IP address), click the Service Associations tab.

3. Regardless of the selection you made in step 2, a dialog box opens. In the **Choose Monitor** box, 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.

To associate a monitor from the command line

Use the **bigpipe node** command to associate a monitor with a node. For example:

```
b node 10.10.10.53:80 monitor use my_http
```

To show or delete associations using the Configuration utility

1. In the navigation pane, click **Monitors**.
   The Monitors screen opens.
2. Click the tab corresponding to the type of association you want:
   - If you are showing or deleting a node association (a node is the IP address plus the service) 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 service association (the service minus the IP address), click the Service Associations tab.
   Regardless of the selection you made in step 2, a dialog box opens showing existing associations.
3. In the **Delete Existing Associations** check box, delete associations by checking the box, and 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, you must 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** command:

```
b node <addr:port> monitor show
b node 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 associations, it is important to consider how the association was made. If a monitor association was created using a wildcard address, the wildcard must be deleted. For example, if multiple associations were created by typing the command **b node *:80 monitor use my_tcp_echo**, you must delete the assocations by typing:

```
b node *:80 monitor delete
```

If multiple associations were created by typing **b node * monitor use my_tcp_echo**, you must delete the assocations by typing:

```
b node * monitor delete
```

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 the monitor 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 you can do 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, 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 Association**s, **Node Association**s, 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 11.26.

```
+- 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 11.26 *All monitor instances disabled*

To disable a monitor instance, use the **bigpipe monitor instance <addr:port> disable** command:

```
b monitor instance <addr:port> disable
```

You can re-enable disabled monitors and instances 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.

12

Filters

- Introducing filters
- IP filters
- Rate filters and rate classes

Introducing filters

Filters control network traffic by specifying whether an external network interface accepts or rejects packets. Filters apply to both incoming and outgoing traffic. When creating a filter, you define criteria that are applied to each packet that the BIG-IP system processes. You can configure the BIG-IP system to accept or block each packet, based on whether or not the packet matches the criteria.

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

Filter options are shown in Table 12.1.

Filter Options	Description
IP filter	You can configure IP filters to control requests sent to the BIG-IP system by other hosts in the network.
Rate filter	You can configure rate filters to control the flow of traffic into the BIG-IP system 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 12.1 *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.*

PART IV
REDUNDANT BIG-IP SYSTEMS

13

Configuring a Redundant System

- Introducing redundant systems
- 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 system to be the preferred active unit
- Setting up active-active redundant BIG-IP units

Introducing redundant systems

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 system 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 system, several advanced redundant features are available. Advanced redundant system features provide additional assurance that your content is available if a BIG-IP system 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 unit to be the active
- Setting up active-active redundant systems

The attributes you can configure for a redundant system are shown in Table 13.1.

Attributes	Description
Synchronizing configurations	This feature allows you to configure one BIG-IP system and then synchronize the configuration with the other BIG-IP system.
Fail-safe for VLANs	Fail-safe for VLANs provides the ability to cause a BIG-IP system 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 system to use the network to determine the status of the active unit.

Table 13.1 *The attributes you can configure for redundant systems*

Attributes	Description
Setting a dominant BIG-IP system	You can set up one unit in a pair to be the dominant active BIG-IP system. The unit you set up as the dominant BIG-IP system always attempts 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 13.1 *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 system:

- The common **bigdb** 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 system 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 **bigdb** keys
- All files in **/config** (except **bigip_base.conf**)

The **bigpipe 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 to memory only on the standby unit, and is therefore not saved.

Configuring fail-safe settings

For maximum reliability, the BIG-IP system supports failure detection on both internal and external VLANs. When you arm the fail-safe option on a VLAN, the BIG-IP system monitors network traffic going through the VLAN. If the BIG-IP system 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 system 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 system is in a stable production environment. Otherwise, routine network changes may cause fail-over unnecessarily.

Each interface card installed on the BIG-IP system 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. Locate the **ArmFailsafe** check box:
   - To arm fail-safe, check the **Arm Failsafe** box.
   - To disarm fail-safe, clear the **Arm Failsafe** box.
4. 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.
5. 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 **timeout** and **failsafe arm** keywords:

```
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 system 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 system 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 system.

Commands for mirroring

Several commands support mirroring capabilities. For complete descriptions, syntax, and usage examples, see Appendix A, *bigpipe Command Reference*.

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 system fails. When you use the mirroring feature, the standby BIG-IP unit 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.

Important

Connection mirroring is not supported on virtual servers that are the targets of an SSL or Akamaizer Proxy.

To control mirroring for a virtual server, use the **bigpipe virtual mirror** command to enable or disable mirroring of connection information. 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 system 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 system provide network failure detection based on network traffic. Gateway fail-safe monitors traffic between the active BIG-IP system 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 using the Configuration utility or within the **bigdb** database. If you configure gateway fail-safe within the **bigdb** database, 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 system 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 the bigdb database

To enable gateway fail-safe in the **bigdb** database, you need to change the settings of three specific **bigdb** 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 system 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 system 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 the **bigdb** database and using **bigpipe db**, see Appendix B, *bigdb 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 global** 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 system console (**/dev/console**).

Using network-based fail-over

Network-based fail-over allows you to configure your redundant BIG-IP system 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.

◆ Note

If VLAN mirroring is configured on the redundant unit, use hard-wired failover instead of network-based failover. Otherise, the redundant unit can lose its state with the peer unit, and traffic on non-mirrored VLANs appears not to function.

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 the bigdb database

To enable network-based fail-over, you need to change the settings of specific **bigdb** 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 system, change the following value using **b db:**

```
b db set Common.Bigip.Cluster.ActiveKeepAliveSec=<value>
b failover init
```

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

Setting a preferred active unit means overlaying the basic behavior of a BIG-IP system with a preference toward being active. A BIG-IP system 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 system in the following description. Each of the two BIG-IP system 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 system is defined by setting the appropriate startup parameters for **the fail-over mechanism** in the **bigdb** database. For more details on **fail-over** startup and functioning, see *Failover and cluster keys*, on page B-2.

To force a BIG-IP unit to active or standby state

The following example shows how to set a BIG-IP unit to standby:

```
b db set Local.Bigip.Failover.ForceStandby
b failover init
```

A BIG-IP unit 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 unit to active:

```
b db set Local.Bigip.Failover.ForceActive
b failover init
```

A BIG-IP unit 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 unit 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 unit assumes the virtual servers of the failed machine. For this configuration to work, each BIG-IP unit 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 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 system.
- 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 **bigdb** 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 the **bigdb** database.
- Synchronize the configuration.
- Complete the transition from active/standby to active-active.

Task 1: Enabling active-active on the BIG-IP system

The first task you need to complete is to enable active-active on the BIG-IP system 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 unit must have its own floating self IP address on the internal VLAN. This is the address to which the servers behind the BIG-IP system 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 system 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 system, 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 unit 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 unit 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 **bigdb** key **Local.Bigip.Failover.UnitId**. This value should be **1** for one of the units, and **2** for the other.

Each BIG-IP system 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 system. 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 system in the Setup utility.

To check the BIG-IP unit number using the Configuration utility

On each BIG-IP unit in a redundant system, check the BIG-IP unit number with the Configuration utility by looking at the upper left corner of the navigation pane. The status of the BIG-IP system 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 system handles connections for the virtual server. Each BIG-IP system 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 system 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 system 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 system 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 system. 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. For more information, see Chapter 10, Address Translation: SNATs, NATs, and IP Forwarding.

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 system 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 the bigdb database

If you change a **bigdb** 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 unit 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 system:

```
b failover standby
```

This command puts the active BIG-IP system into partial active-active mode. To complete the transition, type in the following command on the other BIG-IP unit 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 system****, on page 13-12.*

Understanding active-active system fail-over

Before a failure in an active-active installation, one BIG-IP unit is servicing all requests for virtual servers configured on unit **1**, and the other BIG-IP unit is servicing all requests for virtual servers configured on unit **2**. If one of the BIG-IP units fails, the remaining BIG-IP unit 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 unit is acting as both units **1** and **2**.

If the BIG-IP unit that failed reboots, it re-assumes connections for the unit number with which it was configured. The BIG-IP unit 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 unit 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 **bigdb** key **Common.Bigip.Failover.ManFailBack** to **1**. When you set this key to **1**, a BIG-IP system 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 system 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 system to surrender its unit number to its peer. That is, its peer now becomes both units 1 and 2, the BIG-IP system 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 system 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 system 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 unit to resume its unit number. That is, the peer now relinquishes the unit number of the BIG-IP system 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 system running with both unit numbers.

Introducing additional active-active bigdb configuration parameters

There are several new **bigdb** parameters for active-active mode.

- **Common.Bigip.Failover.ActiveMode**
  Set this **bigdb** 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 system. This value is set by the Setup utility or when you upgrade your units to this version of the BIG-IP system 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 system 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 13-16.
- **Common.Bigip.Failover.AwaitPeerDeadDelay**
  The BIG-IP system 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 system 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 system, and those that are standby connections for the peer BIG-IP system. In prior versions of the BIG-IP system, 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 system. 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 **bigdb** 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 system 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 system to resume normal operation. If manual fail back is enabled, this command causes a BIG-IP system 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 system 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 **bigdb** 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 system 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 unit 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.

PART V
LINK CONFIGURATION

14

Inbound Load Balancing

- Working with load balancing modes for inbound traffic
- Understanding inbound load balancing on the Link Controller
- Configuring inbound load balancing
- Modifying a wide IP

Working with load balancing modes for inbound traffic

The Link Controller uses load balancing modes to distribute DNS name resolution requests on multiple links, sent by local DNS servers, to the best available virtual server in the Link Controller configuration. This is known as ***inbound load balancing.*** This chapter first describes how inbound load balancing works on the Link Controller, explains the various static and dynamic load balancing modes, and then describes how to configure them.

Understanding inbound load balancing on the Link Controller

The Link Controller has an external IP address configured for each network or ISP link that it manages. The external IP addresses are configured as virutal servers on the Link Controller, and these virtual servers belong to a wide IP pool on the Link Controller. The translations for the virtual servers direct the incoming requests to the appropriate content servers (for example, a web server or a database server) on the internal network. This process is known as ***inbound load balancing***.

When the Link Controller receives a name resolution request from a local DNS server, the system uses a load balancing mode to select the best available virtual server in a wide IP. Once the Link Controller selects the best virtual server, it constructs the DNS answer and sends the answer back to the requesting client's local DNS server, using the best available link. The DNS answer, or ***resource record***, is an **A** record that contains one or more virtual server IP addresses.

The Link Controller chooses a virtual server from a wide IP pool using either a static load balancing mode or a dynamic load balancing mode. A ***static load balancing mode*** selects a virtual server based on a pre-defined pattern. A ***dynamic load balancing mode*** selects a virtual server based on current performance metrics.

Within each pool, you specify two different load balancing modes that the system uses in sequential order: the preferred method or the alternate method. The ***preferred*** method is the first load balancing mode that the Link Controller uses for load balancing. If the preferred method fails, the system then uses the **alternate** method for load balancing. If this load balancing mode fails, then the Link Controller randomly picks an available virtual server.

The Link Controller supports several modes for inbound load balancing. Table 14.1 shows a complete list of the modes, and indicates where you can use each mode. The sections following the table describe how each load balancing mode works.

Load Balancing mode	Use for preferred method	Use for alternate method	Static mode	Dynamic mode
Completion Rate	X			X
Global Availability	X	X	X	
Hops	X			X
Kilobytes/Second	X			X
Least Connections	X		X	
Packet Rate	X	X		X
Quality of Service	X			X
Random	X	X	X	
Ratio	X	X	X	
Round Robin	X	X	X	
Round Trip Time	X			X
Static Persist	X	X	X	
VS Capacity	X	X		X

Table 14.1 *Inbound load balancing modes and usage*

Static load balancing modes distribute connections across the network according to predefined patterns, and take server availability into account. ***Dynamic load balancing modes*** distribute connections to servers that show the best current performance. The performance metrics taken into account depend on the particular dynamic mode you are using.

Using static load balancing modes

The Link Controller supports the following static load balancing modes:

- Global Availability
- Random
- Ratio
- Round Robin
- Static Persist

The static load balancing modes perform load balancing, based on predefined criteria, as described in the following sections.

Global Availability mode

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

Random mode

The Random load balancing mode sends connections to virtual servers in a random, uniform distribution pattern. The Random mode is useful for certain test configurations.

Ratio mode

The Ratio load balancing mode distributes connections among a pool of virtual servers as a weighted Round Robin. For example, you can configure the Ratio mode to send twice as many connections to a T1 link, and only half as many connections to a DSL link.

The Ratio load balancing mode requires that you define a ratio weight for each virtual server in a pool. The default ratio weight for a server or a pool is set to **1**. Figure 14.1 shows a sample connection distribution for Ratio mode.

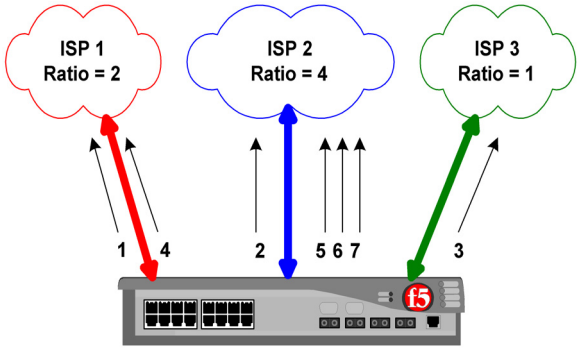

Figure 14.1 *Ratio mode*

Round Robin mode

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

Figure 14.2 shows a sample of the connection distribution pattern for Round Robin mode.

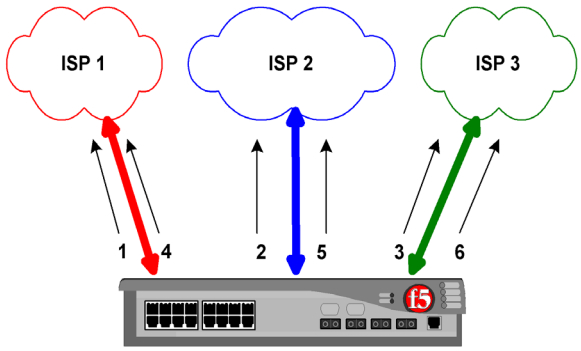

Figure 14.2 *Round Robin mode*

Static Persist mode

The Static Persist load balancing mode provides static persistence of local DNS servers to virtual servers; it consistently maps an LDNS IP address to the same available virtual server for the duration of the session. This mode guarantees that certain transactions are routed through a single transaction manager (for example, a BIG-IP system or other server array manager); this is beneficial for transaction-oriented traffic, such as e-commerce shopping carts, online trading, and online banking.

Using dynamic load balancing modes

All dynamic load balancing modes make load balancing decisions based on the metrics collected by the **big3d** agents running in each data center. The **big3d** agents collect the information at set intervals that you define when you set the global timer variables. If you want to use the dynamic load balancing modes, you must run one or more **big3d** agents in each of your data centers, to collect the required metrics.

The Link Controller supports the following dynamic load balancing modes:

- Completion Rate
- Hops
- Kilobytes/Second
- Least Connections
- Packet Rate
- Round Trip Times (RTT)

- Quality of Service (QOS)
- VS Capacity

Completion Rate mode

The Completion Rate load balancing mode selects the virtual server that currently maintains the least number of dropped or timed-out packets during a transaction between a Link Controller and the client LDNS.

Figure 14.3 shows a sample connection distribution pattern for the Completion Rate mode.

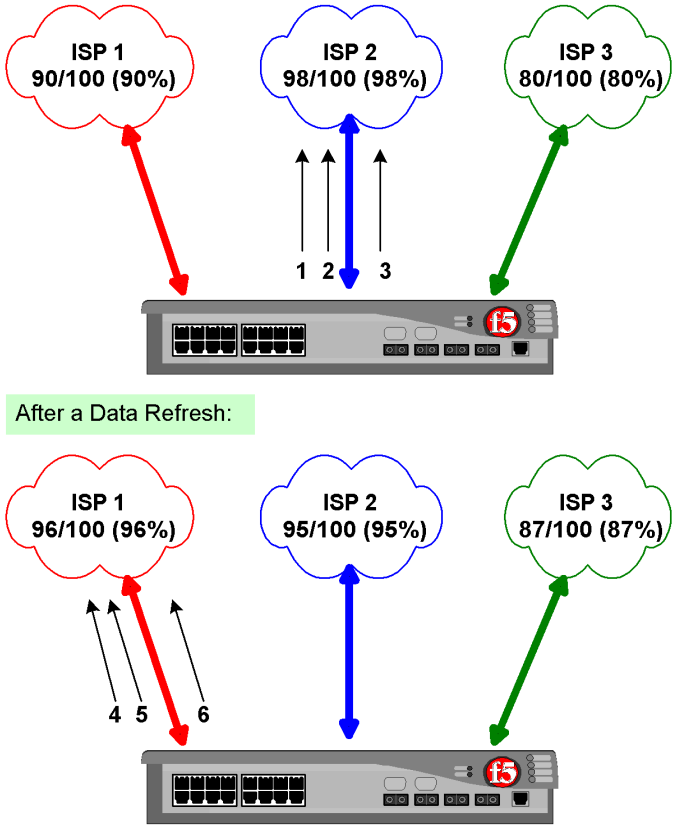

Figure 14.3 *Completion Rate load balancing mode*

Hops mode

The Hops load balancing mode is based on the **traceroute** utility, and tracks the number of intermediate system transitions (router hops) between a client LDNS and each Link Controller. Hops mode selects a virtual server in the data center that has the fewest router hops from the LDNS.

Kilobytes/Second mode

The Kilobytes/Second load balancing mode selects a virtual server that is currently processing the fewest number of kilobytes per second.

Least Connections mode

The Least Connections load balancing mode selects a virtual server on the Link Controller that currently hosts the fewest connections.

Packet Rate mode

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

Figure 14.4 shows a sample connection distribution for the Packet Rate mode.

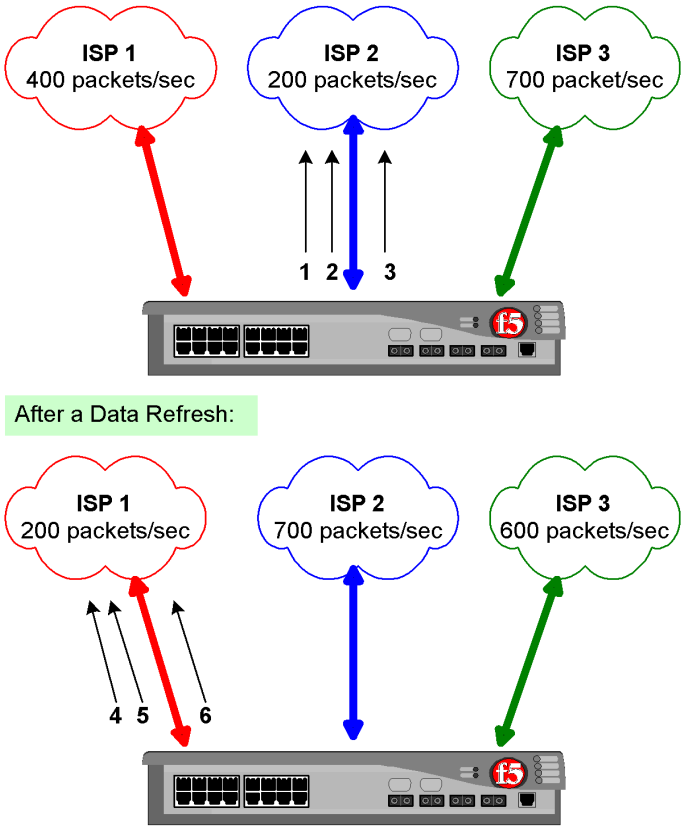

Figure 14.4 *Packet Rate mode*

Round Trip Times mode

The Round Trip Times (RTT) load balancing mode selects the virtual server with the fastest measured round trip time between a Link Controller and a client LDNS.

Figure 14.5 shows a sample connection distribution for the Round Trip Times mode.

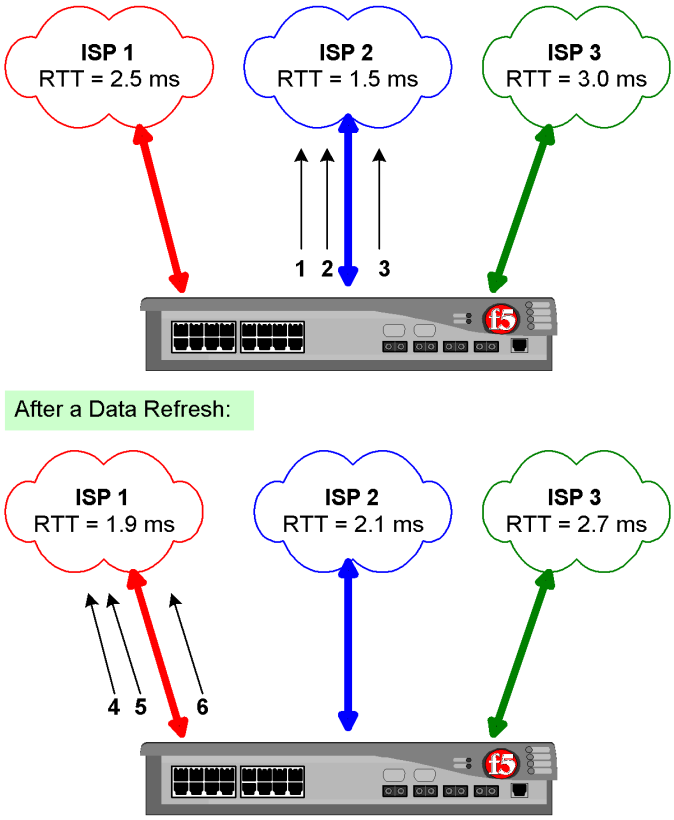

Figure 14.5 *Round Trip Times mode*

Quality of Service mode

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

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

- Link Capacity
- VS Capacity
- Kilobytes/Second

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

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

Understanding QOS coefficients

Table 14.2 lists each Quality of Service (QOS) coefficient, its scale, a likely upper limit for each, and whether a higher or lower value is more efficient.

Coefficient	How measured	Default value	Example upper limit	More efficient setting
Round trip time	Microseconds	50	2,000,000	Lower
Completion rate	Percentage of successfully transferred packets (0-100%)	5	100%	Higher
Hops	Number of intermediate systems transitions (hops)	0	64	Lower
Packet rate	Packets per second	1	700	Lower
Link Capacity	Link bandwidth and cost analysis	30	2,000,000	Lower
VS capacity	Number of nodes ***up***	0	20	Higher
Kilobytes/second	Kilobytes per second throughput	3	15000	Lower

Table 14.2 *QOS coefficients: default values, ranges, and limits*

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

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

- **Normalization**
  The Link Controller normalizes the raw metrics to values in the range of 0 to 10. As the QOS value is calculated, a high measurement for completion rate is good, because a high percentage of completed connections are being made, but a high value for packet rate is not desirable because the packet rate load balancing mode attempts to find a virtual server that is not overly taxed at the moment.
- **Emphasis**
  You can adjust coefficients to emphasize one normalized metric over another. For example, on the Link Controller, the Link Capacity coefficient is set to **1000** by default. If the link capacity scores for two virtual servers are not close, the virtual server with the most available capacity is chosen. If the link capacity scores for both virtual servers are close, the round trip time (RTT) breaks the tie.

Customizing the QOS equation

You can customize the QOS equation for individual wide IPs to tailor Quality of Service load balancing to best match your load balancing requirements.

To modify QOS coefficients using the Configuration utility

1. In the navigation pane, click **Link Configuration**, and then click **Inbound LB**.
   The Wide IPs list screen opens.
2. In the Wide IP column, click a wide IP name.
   The Wide IP Properties screen opens.
3. Click the Wide IP Load Balancing tab.
   The Load Balancing Options screen opens.
4. Modify the QOS coefficients in the **Round Trip Time**, **Completion Rate**, **Hops**, **Packet Rate**, **Link Capacity, VS Capacity**, or **Kilobytes/Second** boxes.
5. Click **Apply**.

Using the Dynamic Ratio option

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

To change the Dynamic Ratio option setting using the Configuration utility

1. In the navigation pane, click **Link Configuration**, and then click **Inbound LB**.
   The Wide IPs list screen opens.
2. In the Wide IP column, click a wide IP name.
   The Wide IP Properties screen opens.
3. Click the Wide IP Load Balancing tab.
   The Load Balancing Options screen opens.
4. To turn the dynamic ratio option off, clear the **Use Dynamic Ratio** check box. To turn the dynamic ratio option on, check the **Use Dynamic Ratio** box. Note that the dynamic ratio option is on by default.
5. Click **Apply**.

VS Capacity mode

The VS Capacity load balancing mode creates a list of the virtual servers, weighted by capacity, then picks one of the virtual servers from the list. The virtual servers with the greatest capacity are picked most often, but over time all virtual servers are returned. If more than one virtual server has the same capacity, then the Link Controller load balances using the Random mode among those virtual servers.

◆ Note

We recommend that you do not use the VS Capacity load balancing mode on the Link Controller because it manages only its own virtual servers.

Configuring inbound load balancing

This section describes how to configure inbound load balancing on the Link Controller. You configure inbound load balancing at the wide IP level.

When you define a wide IP, you specify the preferred and alternate load balancing methods to use in selecting a virtual server within the wide IP pool. You can find instructions on how to configure these load balancing methods in the section, *Defining a wide IP*, following.

Understanding wide IPs

After you configure the virtual servers they manage, you need to group the configured virtual servers into wide IPs. A ***wide IP*** is a mapping of a fully-qualified domain name (FQDN) to a set of virtual servers that host the domain's content, such as a web site, an e-commerce site, or a CDN.

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

- Gather your configuration information for the Link Controller so you can easily see which virtual servers have the content you want to map to a domain name.
- Decide which load balancing modes to use for the wide IP pool.

Understanding wide IP pools

A wide IP contains a pool definition. A ***pool*** is a group of virtual servers that the Link Controller load balances. The virtual servers on a Link Controller are associated with a specific link, and they load balance inbound requests over the links. You should configure a wide IP for each link, so that you can configure a pool that contains that link's virtual servers.

Defining a wide IP

After you determine which virtual servers you should place in which wide IP pool, you are ready to add the first wide IP to the configuration. Note that for the Link Controller, you configure only one pool in the wide IP.

To define a wide IP using the Configuration utility

1. In the navigation pane, click **Link Configuration**, and then click **Inbound LB**.
   The Wide IPs list screen opens.
2. Click the **Add** button.
   The Define Wide IP (Step 1 of 2) screen opens.
3. Add the wide IP name and port settings, and click **Next**.
   The Define Wide IP (Step 2 of 2) screen opens.
4. In the **Available** list, click the virtual servers that you want to add to the pool, and click the Add (-->>) button.
   The virtual servers become part of the **Members** list for the wide IP.
5. Click **Finish**.
   The wide IP is added to your configuration.

Repeat this process for each wide IP you want to add. For help on defining wide IPs and pools, click **Help** on the toolbar.

Using wildcard characters in wide IP names

The Link Controller supports wildcard characters in both wide IP names and wide IP aliases. You can use the wildcard characters to simplify your maintenance tasks if you have a large quantity of wide IP names and/or wide IP aliases. The wildcard characters you can use are: the question mark (**?**), and the asterisk (*****). The guidelines for using the wildcard characters are as follows:

- **The question mark** (**?**)
  - You can use the question mark to replace a single character, with the exception of dots (**.**).
  - You can use more than one question mark in a wide IP name or alias.
  - You can use both the question mark and the asterisk in the same wide IP name or alias.

- **The asterisk** (*****)
  - You can use the asterisk to replace multiple consecutive characters, with the exception of dots (**.**).
  - You can use more than one asterisk in a wide IP name or alias.
  - You can use both the question mark and the asterisk in the same wide IP name or alias.

The following examples are all valid uses of the wildcard characters for the wide IP name, **www.mydomain.net**.

- **???.mydomain.net**
- **www.??domain.net**
- **www.my*.net**
- **www.??*.net**
- **www.my*.***
- **???.my*.***
- ***.*.net**
- **www.*.???**

Modifying a wide IP

Once you have defined the basic settings for a wide IP, you can modify the wide IP settings, the load balancing properties, and the virtual server members at any time.

Modifying the basic wide IP settings

When you first add a wide IP to the configuration, you configure the basic settings. You may later decide to modify those settings.

To modify an existing wide IP using the Configuration utility

1. In the navigation pane, click **Link Configuration**, and then click **Inbound LB**.
   The Wide IPs list screen opens.
2. Click the name of the wide IP that you want to modify.
   The Wide IP Properties screen opens.
3. Make any changes that you want, and click **Apply**. For help on the specific settings, click the **Help** button.

Modifying the load balancing properties

When you first add a wide IP, the load balancing properties are set to a default configuration. You can later change the load balancing modes, adjust the QOS coefficients, set up LDNS Round Robin, or modify the pool time-to-live (TTL).

To modify the load balancing properties for a wide IP using the Configuration utility

1. In the navigation pane, click **Link Configuration**, and then click **Inbound LB**.
   The Wide IPs list screen opens.
2. Click the name of the wide IP that you want to modify.
   The Wide IP Properties screen opens.
3. Click the Wide IP Load Balancing tab.
   The Load Balancing Options screen opens.
4. Make any changes that you want, and click **Apply**. For help on the specific settings, click the **Help** button.

Changing the preferred or alternate load balancing modes

By default, the preferred load balancing mode is set to Quality of Service, and the alternate load balancing mode is set to Round Robin. For details on these load balancing modes as well as the additional load balancing modes, see *Working with load balancing modes for inbound traffic*, on page 14-3.

Using the LDNS round robin wide IP attribute

LDNS round robin is an attribute that you can use in conjunction with any load balancing mode. The LDNS round robin attribute allows the Link Controller to return a list of available virtual servers, instead of a single

virtual server. Certain browsers keep the answer returned by DNS servers. By enabling this attribute, the Link Controller returns a maximum of 16 virtual servers as the answer to a DNS resolution request. This provides browsers with alternate answers if a virtual server becomes unavailable.

Adjusting the QOS coefficients

You can adjust the QOS coefficients to best match your requirements. By default, only the Link Capacity coefficient has a value. For more information on working with the QOS coefficients, see *Understanding QOS coefficients*, on page 14-10.

15

Internet Link Evaluator

- Overview of the Internet Link Evaluator
- Interpreting the Internet Link Evaluator data

Overview of the Internet Link Evaluator

The Internet Link Evaluator, a statistics screen in the Configuration utility, is a unique tool with which to evaluate the following data about the Internet:

- The average round trip time between the local DNS servers on a particular continent and the links in your network
- The average completion rate between the local DNS servers on a particular continent and the links in your network
- The average number of router hops between the local DNS servers on a particular continent and the links in your network

The data displayed in the Internet Link Evaluator is based on path data, which is collected when you use a dynamic load balancing mode, such as Round Trip Times or Quality of Service. For more information on dynamic load balancing modes, see *Using dynamic load balancing modes*, on page 14-6.

To view the Internet Link Evaluator statistics screen using the Configuration utility

1. In the navigation pane, expand the **Link Statistics** item.
2. Click **Link Evaluator**.
   The Internet Link Evaluator Statistics screen opens.
3. For information on working with the Internet Link Evaluator Statistics screen, view the online help.

The round trip time and completion rate data, on the Internet Link Evaluator Statistics screen, are based on path data. The router hops data is based on information collected by the **traceroute** utility.

Working with the Average Round Trip Time table

In the Average Round Trip Time table on the Internet Link Evaluator Statistics screen, you can view the following information:

- The average round trip time for each link to each continent
- For each link, the best average round trip time to the local DNS servers on a particular continent. This value is indicated by **bold** text within the table.
- For each continent, the best average round trip time from the links. This value is indicated by underlined text within the table.

If you hold the mouse pointer over the Information button ( ), you can view the following additional information:

- For a particular link, the number of local DNS servers used to calculate the average round trip time

- For all the local DNS servers that have been probed by a particular link, the percentage of those local DNS servers that are located on a particular continent
- For all the local DNS servers on a particular continent, the percentage of those local DNS servers that have been probed by a particular link

Working with the Average Completion Rate table

In the Average Completion Rate table on the Internet Link Evaluator Statistics screen, you can view the following information:

- The average completion rate for each link to each continent
- For each link, the best average completion rate to the local DNS servers on a particular continent. This value is indicated by **bold** text within the table.
- For each continent, the best average completion rate from the links. This value is indicated by underlined text within the table.

If you hold the mouse pointer over the Information button ( ), you can view the following additional information:

- For a particular link, the number of local DNS servers used to calculate the average completion rate
- For all the local DNS servers that have been probed by a particular link, the percentage of those local DNS servers that are located on a particular continent
- For all the local DNS servers on a particular continent, the percentage of those local DNS servers that have been probed by a particular link

Working with the Average Router Hops table

In the Average Router Hops table on the Internet Link Evaluator Statistics screen, you can view the following information:

- The average number of router hops between each link and each continent
- For each link, the best average number of router hops to the local DNS servers on a particular continent. This value is indicated by **bold** text within the table.
- For each continent, the best average number of router hops from the links. This value is indicated by underlined text within the table.

If you hold the mouse pointer over the Information button ( ), you can view the following additional information:

- For a particular link, the number of local DNS servers used to calculate the average number of router hops
- For all the local DNS servers that have been probed by a particular link, the percentage of those local DNS servers that are located on a particular continent

- For all the local DNS servers on a particular continent, the percentage of those local DNS servers that have been probed by a particular link

Interpreting the Internet Link Evaluator data

In choosing an ISP provider, enterprises have very few ways to evaluate the quality of the connectivity provided. By measuring the round trip times, completion rate, and average router hops over the configured links on the Link Controller, the Internet Link Evaluator provides a thorough view of link and ISP performance. By comparing link performance over time, you can monitor the actual performance that an ISP delivers to your customers.

The two data points that help you determine which link has the best performance are the RTT response time (lower is better), and the Completion Rate (higher is better). The Round Trip Time measurement calculates the packet rate of a request from each geographically-distributed user. The Completion Rate measurement is an important metric for evaluating the quality of a link, because it represents the percentage of dropped packets. Completion rate is also important for measuring line quality for applications such as streaming media.

◆ Note

One easy way to compare link performance over time is to print a screen shot of the Internet Link Evaluator at a certain time every day.

16

Working with Link Configuration

- Overview of link configuration
- Viewing link statistics and metrics

Overview of link configuration

Link configuration is based on entries you make when you run the Setup utility. When you run the Setup utility, you add the members for the default gateway pool, the self IP addresses for the Link Controller, and the VLANs. These objects are the basic components for link configuration.

You configure the links that you want to load balance in the Configuration utility. When you initially add links, you configure the basic properties for the link. Once you have configured the basic properties, you can refine and enhance the link configuration with several advanced configuration options. This chapter explains how to configure the basic properties of a link, and also how to configure the advanced properties of a link.

◆ Note

*For specific examples of how to deploy a link load balancing solution in your network, refer to the **Link Controller Solutions Guide**.*

Defining the basic properties for a link

Before you can load balance inbound and outbound traffic on the Link Controller, you configure the basic links. The following procedure describes how to configure the basic properties of a link.

To configure the basic link properties using the Configuration utility

1. In the navigation pane, expand the **Link Configuration** item, and and then click **Links**.
   The Link List screen opens.
2. Click the **Add** button.
   The Link Configuration: Step 1 of 2 screen opens.
3. In the **Link Name** box, type a name for the link.
4. In the **Self IP** box, select a self IP address to associate with the link.
5. In the **VLAN** box, type the name of the VLAN associated with the self IP address you selected in the previous step.
6. Next to the **Gateway IP** box, click the **Add** button to select the gateway IP address to associate with the link.
7. If you want to limit the volume of inbound traffic, check the box next to **Inbound Traffic Limit**, and type a number that represents the maximum throughput rate (in kilobits per second) for the inbound traffic on the link. (Note that this is an optional setting.)
8. If you want to limit the volume of outbound traffic, check the box next to **Outbound Traffic Limit**, and type a number that represents the maximum throughput rate (in kilobits per second) for the outbound traffic on the link. (Note that this is an optional setting.)

9. If you want to limit the total volume of bidirectional traffic, check the box next to **Total Traffic Limit**, and type a number that represents the maximum throughput rate (in kilobits per second) for the sum of inbound and outbound traffic on the link. (Note that this is an optional setting.)
10. Click **Next**.
    The Link Configuration (Step 2 of 2) screen opens.
11. In the **Monitor Name** box, type a name for the health monitor for the link. Alternately, you can select a default monitor from the list. (Note that this is an optional setting. If you do not want to configure a monitor at this time, skip to step 13.)
12. In the **Monitor Type** box, select the type of monitor for the link.
13. Click **Finish**.
    The link and optional monitor are added to the configuration, and the Link List screen opens.

◆ Note

*For details on any of the settings on the two screens in the Link Configuration wizard, click the **Help** button.*

Working with the advanced properties for a link

Once you have configured the basic link properties, you can configure several advanced properties for the link. The advanced properties include:

- Additional basic link properties
- Additional health monitor properties
- Link weighting and billing properties
- SNMP properties

Configuring additional basic link properties

Once you have configured the basic properties of a link, you can adjust those properties, and configure several advanced properties. This procedure describes how to configure additional link properties.

To configure additional link properties using the Configuration utility

1. In the navigation pane, expand the **Link Configuration** item, and then click **Links**.
   The Link List screen opens.
2. Click the name of the link that you want to modify.
   The Link Properties tab opens.

3. Make any changes that you want to the basic properties of the link. In addition to the properties that you defined when you first configured the link, you can add an ISP name, or redefine the link bandwidth limit settings. For details on the additional settings on this screen, click the **Help** button.
4. When you have completed the changes that you want to make, click **Apply.**

Configuring additional health monitor properties

You can configure a health monitor when you first configure a link, or you can configure a health monitor at a later time. You can also configure additional settings for an existing monitor.

To configure additional health monitor properties using the Configuration utility

1. In the navigation pane, expand the **Link Configuration** item, and then click **Links**.
   The Link List screen opens.
2. Click the name of the link that you want to modify.
   The Link Properties tab opens.
3. Click the Link Monitor tab.
   The Monitor screen opens.
4. On the Monitor screen, you can define a monitor if you did not define one when you first configured the link. You can also configure additional settings for an existing monitor, for example **Timeout** and **Interval**. For details on the additional settings on this screen, click the **Help** button.
5. When you have completed the changes that you want to make, click **Apply**.

◆ Note

For additional information about health monitors, see Chapter 11, Monitors.

Configuring link weighting and billing properties

With the link weighting and billing properties, you can refine the link load balancing and statistics reporting for the links in your configuration. On the Link Weighting screen, you determine how the Link Controller manages and distributes traffic across the links, using the following settings.

- **Duplex Billing**
  If your ISP provider uses duplex billing, you can set the **Duplex Billing** option so that the statistics and billing report screens on the Link Controller accurately reflect the bandwidth usage for the link. Note that by default, the duplex billing setting is on.
- **Price Weighting**
  If you want the Link Controller to load balance traffic based on the cost of the bandwidth, then select the **Use Price Weighting** option. You can use this weighting option to avoid the costs associated with exceeding your prepaid bandwidth. You can also use this weighting option to direct traffic over the least expensive link first.
- **Ratio Weighting**
  If you want the Link Controller to load balance the total traffic to the controller based on a ratio, then select the **Use Ratio Weighting** option. When you have links of varying bandwidth sizes, you can use this weighting option to avoid oversaturating a smaller link with too much traffic.

Important

You can use either the price weighting option or the ratio weighting option to load balance your link traffic for all of your links. You cannot use both options. Regardless of which weighting option you use, you must use the same weighting option for all links.

To configure link weighting and billing properties using the Configuration utility

1. In the navigation pane, expand the **Link Configuration** item, and then click **Links**.
   The Link List screen opens.
2. Click the name of the link that you want to modify.
   The Link Properties tab opens.
3. Click the Link Weighting tab.
   The Link Weighting screen opens.
4. Make any changes that you want for the link weighting and billing properties. For details on the settings on this screen, click the **Help** button.
5. When you have completed the changes that you want to make, click **Apply**v.

Configuring SNMP properties

If you want the Link Controller to report the metrics gathered directly from the gateway router for the link, you need to enable SNMP for the link. Use the following procedure to enable SNMP communications with the gateway router.

◆ Note

You must also enable SNMP communications on the router itself. Refer to the router manufacturer's documentation for instructions.

To enable SNMP for the gateway router using the Configuration utility

1. In the navigation pane, expand the **Link Configuration** item, and then click **Links**.
   The Link List screen opens.
2. Click the name of the link that you want to modify.
   The Link Properties tab opens.
3. Click the Link SNMP tab.
   The SNMP screen opens.
4. Add the settings to enable SNMP communications for the router. For details on the settings on this screen, click the **Help** button.
5. When you have completed the changes that you want to make, click **Apply**.

Viewing link statistics and metrics

Using the Configuration utility, you can view current statistics about the following objects in the Link Controller configuration:

Statistics screen	Description
Summary	Provides information about the Link Controller itself.
Metrics	Provides performance information for the routers, virtual servers, and pools you have configured.
Links	Provides information about the links in the network.
Disabled	Provides information on the links, virtual servers, wide IPs, and pools that are currently disabled.
Requests	Provides information on the virtual connections between local DNS servers and virtual servers for given wide IPs in the network.
Wide IPs	Provides information on the wide IPs, pools, and the virtual servers in the pools.
Virtual servers	Provides information on the virtual servers you have configured.
P95 Billing	Provides information about the average actual link utilization compared to purchased bandwidth.
Link Evaluator	Provides information on the average round trip times, average completion rates, and average router hops between the data centers or links you have configured and local DNS servers.
Paths	Provides information on the paths created by the Link Controller when paths are required to fulfill name resolution requests.
Local DNS	Provides information on the local DNS servers in the Link Controller database.
Config Checker	Provides information about your link configuration. When you are setting up your Link Controller, you can use the Configuration Checker to verify the setup.

Table 16.1 *Link Statistics screens*

To view the Link Statistics screens

1. In the navigation pane, expand the **Link Statistics** item, and then click the statistics screen that you want to review.
2. For details on the information provided on each screen, click **Help**.

PART VI
BIG-IP SYSTEM ADMINISTRATION

17

Administering the BIG-IP System

- Monitoring and administration utilities
- Using the bigpipe utility as a monitoring tool
- Customizing the Configuration utility user interface
- Working with the bigtop utility
- Working with the Syslog utility
- Powering down the BIG-IP system
- Removing and returning items to service
- Viewing system statistics and log files
- Managing user accounts
- Working with the bigdb database

Monitoring and administration utilities

The BIG-IP system provides several utilities for monitoring and administering the BIG-IP system.

- **The Configuration utility**
  You can use the Configuration utility to manage any feature on the BIG-IP system. For example, you can reset any statistic, or all statistics, for virtual servers, nodes, NATs, and SNATs, using the Configuration utility.
- **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 system.
- **bigtop**
  The bigtop utility 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 that monitors critical system events, as well as configuration changes made on the BIG-IP system.
- **bigdb**
  bigdb is a database that contains various configuration information for the BIG-IP system.

Using the bigpipe utility as a monitoring tool

Using the **bigpipe** utility, you can view information about the BIG-IP system 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 system

The **bigpipe summary** command displays performance statistics for the BIG-IP system itself. These statistics can be used to monitor and troubleshoot your BIG-IP system.

The output of the **bigpipe summary** command includes current usage statistics, such as the amount of time a BIG-IP system has been running since the last reboot. To display a summary of the performance statistics for the BIG-IP system, type the following command:

```
b summary
```

If you need to reboot the BIG-IP system but want to retain the current summary information, you can save the information in a file prior to rebooting. For more information, see *Retaining current statistics during reboot*, on page 17-10.

The performance statistics display in the format shown in Figure 17.1 (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
BIG-IP no handler deny                   = 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 17.1 *The* ***bigpipe summary*** *display screen*

Explanation of summary statistics

Table 17.1 contains descriptions of each individual statistic included in the **bigpipe summary** display screen.

Statistic	Description
total uptime	Total time elapsed since the BIG-IP system was last booted. Output is displayed in days, hours, minutes, and seconds.
total uptime (secs)	Total uptime since the last reboot, displayed in seconds.
total # connections	Total number of connections handled by the BIG-IP system since the last reboot. In this case, a connection is a data path between a client machine, a virtual server, NAT or SNAT, and a node. Connections last until timed out by either a reaper or UDP timer. This is important to note, because a BIG-IP system with short time-outs is likely to show more connections than a BIG-IP system with long time-outs.
total # pkts	Total number of packets handled by the BIG-IP system since the last reboot. This value equals the sum of **total # pkts (inbound)** and **total # pkts (outbound)**. By dividing this value by **total uptime (secs)**, you can determine the average packets per second since the last reboot.
total # bits	Total number of bits handled by the BIG-IP system since the last reboot. This value should equal the sum of **total # of bits (inbound)** and **total # bits (outbound)**. By dividing this value by **total uptime (secs)**, you can determine the average bits per second since the last reboot.
total # pkts (inbound)	Total number of incoming packets handled by the BIG-IP system since the last reboot.
total # bits (inbound)	Total number of incoming bits handled by the BIG-IP system since the last reboot.
total # pkts (outbound)	Total number of outgoing packets handled by the BIG-IP system since the last reboot. Outgoing packets are packets that originated at the BIG-IP system.
total # bits (outbound)	Total number of outgoing bits handled by the BIG-IP system since the last reboot.

error no nodes available	The number of times the BIG-IP system tried to make a connection to a node, but no nodes were available. Because no nodes were available, the incoming packet was dropped. This is a serious error. If this number is non-zero, then one or more of the virtual servers is, or has been, unavailable.
	For example, the following shows sample statistics for sytems A, B, and C.
	`System A: BIG-IP error no nodes available = 0`
	`System B: BIG-IP error no nodes available = 11409162`
	`System C: BIG-IP error no nodes available = 42`
	In this example, the statisitcs show that System A is healthy. System B, however, is unhealthy; in slightly over a month of **up** time, 11,000,000 instances of an unavailable server have occurred. System C is also unhealthy; although the number of times a virtual server has become unavailable is minimal, this averages to one such event per day of **up** time on this system. It should be possible to keep this number to **zero**.

***Table 17.1 bigpipe** monitoring statistics*

Statistic	Description

tcp port deny	The number of times a client attempted to connect to an unauthorized TCP port on the BIG-IP system. This statistic indicates that a connection was attempted, but the TCP port requested was not open. Because the port was not open, the incoming packet was discarded. This applies only to administrative and alias addresses on the BIG-IP system, and not to virtual servers, NATs, or SNATs.
	If the **sysctl bigip.verbose_log_level** is set to **2** or **15**, the offending port and source IP address are logged in the file **/var/log/bigip**. Alternatively, the location of this log can be set in the **Advanced Properties** section of the browser interface.
	The following shows sample statistics for systems A, B, and C:
	`System A: BIG-IP tcp port deny = 0`
	`System B: BIG-IP tcp port deny = 16358`
	`System C: BIG-IP tcp port deny = 58`
	In these examples, there is a wide disparity in port denials from system to system. System A is behind a firewall and has been available for a relatively short amount of time.
	System B is not behind a firewall, and because of this, the system is intercepting many packets that would otherwise be filtered out by a firewall.
	System C is behind a firewall, but a few packets appear not to have been discarded. In this case, the administrator should examine the BIG-IP system logs to determine the port and source address of these packets and then filter them out at the firewall.

udp port deny	The number of times a client attempted to connect to an unauthorized UDP port on the BIG-IP system. This statistic indicates that a connection was attempted, but the UDP port requested was not open. Because the port was not open, the incoming packet was discarded. This applies only to administrative and alias addresses on the BIG-IP system, and not to virtual servers, NATs, or SNATs.
	If the **sysctl bigip.verbose_log_level** is set to **1** or **15**, the offending port and source IP address are logged in the file **/var/log/bigip**. Alternatively, the location of this log can be set in the **Advanced Properties** section of the browser interface.
	For example, the following shows sample statistics for systems A, B, and C:
	`System A: BIG-IP udp port deny = 0`
	`System B: BIG-IP udp port deny = 16358`
	`System C: BIG-IP udp port deny = 58`
	In these examples, there is a wide disparity in port denials from system to system. System A is behind a firewall and has been available for a relatively short amount of time.
	System B is not behind a firewall, and because of this, the system is intercepting many packets that would otherwise be filtered out by a firewall.
	System C is behind a firewall, but a few packets appear not to have been discarded. In this case, the administrator should examine the BIG-IP system logs to determine the port and source address of these packets and then filter them out at the firewall.

***Table 17.1 bigpipe** monitoring statistics*

Statistic	Description

virtual tcp port deny	The number of times a client attempted to connect to an unauthorized TCP port on a virtual IP address. This statistic indicates that a connection was attempted to a virtual IP address, but used a TCP port that was not open. Because the port was not enabled for this virtual server, the incoming packet was discarded.
	If the **sysctl bigip.verbose_log_level** is set to **8** or **15**, the offending port and source IP address are logged in the file **/var/log/bigip**. Alternatively, the location of this log can be set in the **Advanced Properties** section of the browser interface.
	For example, the following shows sample statistics for systems A, B, and C:
	`System A: BIG-IP virtual tcp port deny = 67`
	`System B: BIG-IP virtual tcp port deny = 221313359`
	`System C: BIG-IP virtual tcp port deny = 447`
	This is another statistic where a critical factor is whether or not the BIG-IP system is located behind a firewall. Systems A and C are placed behind a firewall, while System B is not.
	Because System B is not behind a firewall, it has had to discard many more stray packets (over 221,000,000) than the systems protected by a firewall.
	Although Systems A and C are located behind firewalls, some packets are still slipping through. This indicates that the firewall administrator has probably allowed all ports for each virtual IP address. By tightening security on the firewalls, it should be possible to reduce this statistic to zero. This example applies also to the next section.

virtual udp port deny	The number of times a client attempted to connect to an unauthorized UDP port on a virtual IP address. This statistic indicates that a connection was attempted to a virtual IP address, but used a TCP port that was not open. Because the port was not enabled for this virtual server, the incoming packet was discarded.
	If the **sysctl bigip.verbose_log_level** is set to **4** or **15**, the offending port and source IP address are logged in the file **/var/log/bigip**. Alternatively, the location of this log can be set in the **Advanced Properties** section of the browser interface.
	For example, the following shows sample statistics for systems A, B, and C:
	`System A: BIG-IP virtual tcp port deny = 67`
	`System B: BIG-IP virtual tcp port deny = 221313359`
	`System C: BIG-IP virtual tcp port deny = 447`
	This is another statistic where a critical factor is whether or not the BIG-IP system is located behind a firewall. Systems A and C are placed behind a firewall, while System B is not.
	Because System B is not behind a firewall, it has had to discard many more stray packets (over 221 million) than the systems protected by a firewall.
	Although Systems A and C are located behind firewalls, some packets are still slipping through. This indicates that the firewall administrator has probably allowed all ports for each virtual IP address. By tightening security on the firewalls, it should be possible to reduce this statistic to zero.

max connections deny	The total number of connections denied because the maximum number of connections allowed was exceeded. This error indicates that the maximum number of connections was received on a port that has a connection limit configured. If port connection limits are configured, it is a good idea to monitor this statistic. The statistic indicates the exact amount of traffic being discarded by the port limit. You can set port connection limits using the following **bigpipe service** syntax: `bigpipe service <port> limit <seconds>`

***Table 17.1 bigpipe** monitoring statistics*

Statistic	Description
virtual duplicate syn ssl	This error is obsolete and therefore should always be zero.

virtual duplicate syn wrong dest	The number of duplicate connection attempts from the same client (IP address and port combination) to a different virtual server. This error indicates that a connection arrived from a source port and IP address for which a connection already existed on the BIG-IP system; however, this connection requested a different destination. The existing connection was torn down and a new connection was created.
	This is most likely to occur with packets coming from busy proxies where source ports are constantly being reused. These errors are common and should not be a cause for alarm.
	For example, the following shows sample statistics for systems A, B, and C:
	`System A: BIG-IP virtual duplicate syn wrong dest = 0`
	`System B: BIG-IP virtual duplicate syn wrong dest = 2673815`
	`System C: BIG-IP virtual duplicate syn wrong dest = 324`
	In this example, System A has probably not encountered this error due to its limited **up** time.
	System B has a very large value because the HTML contains links to another virtual server on the BIG-IP system. When these links are followed and coincide with reuse of a source port, the error is generated.
	System C has a fairly average value. It is receiving some source port reuse, probably from large proxies such as AOL.

virtual duplicate syn node down	The number of duplicate connection attempts to a server that is unavailable when a connection to the server was made previously. This error indicates that a **syn** packet arrived and that a connection to a node was created. Later, another **syn** packet arrived but the node was unavailable. The existing connection was severed and a new connection to a different node was created to service the request.
	This error is common when nodes are unreliable or over-burdened.
	The seriousness of this error varies from site to site. In sites where there is a large number of low-capacity nodes, you see this error as nodes become over-burdened and then unavailable. This error is also more common with some load balancing modes, where certain nodes receive more connections than others.
	For example, the following shows sample statistics for systems A, B, and C:
	`System A: BIG-IP virtual duplicate syn node down = 0`
	`System B: BIG-IP virtual duplicate syn node down = 7442773`
	`System C: BIG-IP virtual duplicate syn node down = 1`
	In this example, Systems A and C are healthy, while System B appears to have a serious problem. System B nodesappear to be frequently marked unavailable.

virtual maint mode deny	The number of times a connection to a virtual server was denied while the BIG-IP system was in maintenance mode. Current maintenance status can be determined by using the **bigpipe maint** command.
virtual addr max connections deny	The number of virtual IP address connections that were dropped because the maximum number of connections configured was exceeded. A connection limit on a virtual IP address can be configured using the **bigpipe virtual** command, as shown in the following example: `bigpipe virtual 172.16.1.1 limit 100`

***Table 17.1 bigpipe** monitoring statistics*

Statistic	Description
virtual path max connections deny	The number of virtual path connections that were dropped because the maximum number of connections configured was exceeded. A connection limit on a virtual server can be configured using the **bigpipe virtual** command, as shown in the following example: `bigpipe virtual 172.16.1.1:80 limit 100`

virtual non syn	The number of packets received that are not connection requests, and are destined to a virtual IP address but not a valid virtual server (port). This error indicates that a packet arrived with a destination of a virtual server. However, this packet was not a **syn** and there was no existing connection. The packet was discarded because BIG-IP system only creates a new connection to a virtual server when it receives a **syn** packet.
	This error is common and could occur for a variety of reasons.
	It is typical for this number to grow very large. In most cases, this error occurs when packets arrive out of sequence or when packets are dropped before arriving at the BIG-IP system.
	For example, the following shows sample statistics for systems A, B, and C:
	`System A: BIG-IP virtual non syn = 2`
	`System B: BIG-IP virtual non syn = 1609400253`
	`System C: BIG-IP virtual non syn = 10366`
	In this example, System A and System C show fairly typical values based on their **up** times.
	System B shows a very large value which may indicate a problem. One possibility is that System B has a heavy traffic load and that a significant portion of this traffic must traverse a very long distance before arrival (European traffic to the US west coast, for example). Long physical distance seems to contribute to the kind of packet loss that results in this error. Another possibility is that a device on the local network is dropping packets. Large values in this statistic should be investigated at least enough to ensure that a local device is not the cause of the problem.

no handler deny	The number of packets from the middle of a flow that do not match any known virtual servers or entries in the connection table.
error virtual fragment no port	The number of IP fragments for which there is no port. This error indicates that a packet fragment arrived, but it was not the first fragment. Since the first fragment contains the port number and is required to set up a new connection, BIG-IP system discards the fragment. This error is similar to **virtual non syn**, but refers to fragmented, rather than intact, packets. As with **virtual non syn**, this probably occurs because fragments arrive out of sequence or not at all. This is a common error and not usually a cause for concern.
error virtual fragment no conn	The number of IP fragments for which there is no connection. This error indicates that a packet fragment arrived, but it was not the first fragment. BIG-IP system previously received the first fragment, but it was not a **syn** packet and did not match an existing connection, causing it to be discarded. This error is most likely to occur in conjunction with a **virtual non syn**. Thus, when BIG-IP system received the first packet, it found that the packet met the criteria for **virtual non syn** and dropped it; this error simply indicates that the BIG-IP system is dropping the following fragments of that packet. As with **virtual non syn** and **virtual fragment no port,** this error is relatively common and should not cause much concern.

***Table 17.1 bigpipe** monitoring statistics*

Statistic	Description
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. This error indicates that a packet arrived at the standby BIG-IP system with a destination of the shared alias. Because the BIG-IP system was in standby mode, this packet was dropped. This is most likely to occur when a BIG-IP system has failed over, but the **arp** cache of another device has not yet been updated. Because that device is caching an invalid **arp** request, the device continues to send packets to the previously active BIG-IP system.
dropped inbound	The total number of inbound packets dropped by the BIG-IP system because the source IP address is other than a BIG-IP system address.
dropped outbound	The total number of outbound packets dropped by the BIG-IP system because the source IP address is the BIG-IP system address.
reaped	The total number of TCP connections that timed-out and were therefore deleted by the BIG-IP system.
ssl reaped	The total number of SSL persistence records based on SSL session IDs that timed-out and were therefore deleted by the BIG-IP system. The timeout for SSL persistence is configured when creating the virtual server.
persist reaped	The total number of non-SSL persistence records that timed-out and were therefore deleted by the BIG-IP system. Persistence timeouts are configured when creating the virtual server.
udp reaped	The total number of UDP connections that timed-out and were therefore deleted by the UDP timer of the BIG-IP system. UDP timers are configured with the **bigpipe** command, using the following syntax: `bigpipe udp <port> <timeout>`
malloc errors	The number of times a connection could not be created due to insufficient memory. If this error is occuring, the BIG-IP system is overloaded and the configuration should either be streamlined, or the software and/or chassis should be upgraded.
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 system.
mem percent used	The total percentage of memory in use by all combined memory pools.

***Table 17.1 bigpipe** monitoring statistics*

Retaining current statistics during reboot

When you reboot the BIG-IP system, all existing summary statistics are deleted. If you need to reboot the BIG-IP system but want to save the existing summary statistics, you can a create shell script that saves the statistics and copies them to another file. The following shows an example of this script, named **restart**. Note that this script must reside in the **/sbin** directory.

```
# restart script
bigpipe summary >/var/log/summary.`date +%d%m%y%H%M`
bigpipe ms >/var/log/ms.`date +%d%m%y%H%M`
echo "Reboot by /sbin/restart at `date +%d%m%y%H%M`" >/var/log/bigip
/bin/sync;/bin/sync
/sbin/reboot
```

Figure 17.2 *Script for saving summary statistics*

Resetting statistics on the BIG-IP system

With the **bigpipe** commands, you can selectively reset any statistic on the BIG-IP system. 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 global 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

The following command resets 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 command:

```
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 command to reset statistics for all virtual ports:

```
b service stats reset
```

Use the following command 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 virtual ports **23**, **80**, and **443**, use the following command:

```
b service 23 80 443 stats reset
```

To reset statistics for network address translations (NATs)

Use the following command 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 origin IP addresses **172.20.3.101** and **172.20.3.102**, 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 command 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 SNAT origin addresses **172.20.3.101** and **172.20.3.102**, 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 using 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.

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.

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 system.

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 system displays the statistics as shown in Figure 17.3.

```
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 17.3 *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 system. 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 system. 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 system displays the statistical information as shown in Figure 17.4.

```
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 17.4 *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 system. 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 17.5.

```
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 17.5 *NAT statistics screen*

Monitoring SNATs

The **bigpipe snat show** command displays the status of the SNATs configured on the BIG-IP system. 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
```

Customizing the Configuration utility user interface

You can customize the appearance of the Configuration utility.

To customize the Configuration utility user interface

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.

Working with the bigtop utility

The bigtop™ utility 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. The bigtop utility accepts options that 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 bigtop utility displays the statistics as shown in Figure 17.6.

```
                   |  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 17.6 *The bigtop screen display*

Using bigtop command options

The syntax for the the **bigtop** command is as follows:

```
bigtop  [options...]
```

Table 17.2 lists and describes the options you can use with the **bigtop** command.

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 bigtop 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.
-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 17.2 *bigtop command options*

Using runtime commands in bigtop

Unless you specified the **-once** option, the bigtop 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 bigtop utility.

Working with the Syslog utility

The BIG-IP system 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 system 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 17.3 shows sample log messages to give you an idea of how the Syslog utility tracks events that are specific to the BIG-IP system.

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 system.
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 17.3 *Sample Syslog messages*

Powering down the BIG-IP system

If you want to power down, or turn off, the BIG-IP system, you need to complete two tasks. The first task is to shut down the BIG-IP system software. After you shut down the BIG-IP system software, you can turn the off the power to the system.

To shut down the BIG-IP software from the command line

To complete the first task to shut down the BIG-IP system 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 system, 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 system allows the node to go down only after all the current connections are complete. During this time, the BIG-IP system does not attempt to send new connections to the node. Although the BIG-IP system 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 system 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 system from service

The BIG-IP system offers a Maintenance mode, which allows you to remove the BIG-IP system 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 system no longer accepts connections to the virtual servers it manages. However, it allows the existing connections to finish processing so that current clients are not interrupted.

The **bigpipe maint** command toggles the BIG-IP system into or out of Maintenance mode. Use the following command to put the BIG-IP system in maintenance mode:

```
b maint
```

If the BIG-IP system runs in Maintenance mode for less than 20 minutes and you return the machine to the normal service, the BIG-IP system quickly begins accepting connections. However, if the BIG-IP system 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 system also supports taking only selected virtual servers, addresses, or ports out of service, rather than removing the BIG-IP system 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, use the appropriate command syntax:

```
b virtual <virtual addr>:<virtual port> enable
b virtual <virtual addr>:<virtual port> disable
```

To enable or disable a virtual address, use the appropriate command syntax:

```
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 command 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**, use the appropriate command syntax:

```
b node <node addr>:<node port> enable
b node <node addr>:<node port> disable
```

To enable or disable a **node address**, use the appropriate command syntax:

```
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 more information about using **bigpipe** commands on the BIG-IP system, see Appendix A, *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 system events, such as defining a virtual server
- The Pinger log, which displays status information determined by each node ping issued by the BIG-IP system

Managing user accounts

When you run the Setup utility for the first time to configure your base network, the BIG-IP system automatically creates two special user accounts--**root** and **admin**. As an option, you can also specify within the Setup utility that you want the BIG-IP system to create a third account, **support**, which gives F5 Networks support personnel access to your system. For information on using the Setup utility to create the **root**, **admin**, and **support** accounts, see Chapter 2, *Using the Setup Utility*.

Once the Setup utility has created these accounts, you will most likely want to create additional administrative accounts and assign various system access levels, or user roles, to them, on an ongoing basis.

The remainder of this chapter addresses the following topics:

- Understanding user roles
- Creating and authorizing local user accounts
- Creating and authorizing remote user accounts
- Managing passwords for local accounts
- Managing system accounts

Understanding user roles

Users that have user roles assigned to them fall into one of two categories--fully-privileged users or restricted users. The following sections describe these user-role categories.

Fully-privileged users

Fully-privileged users are those who have full access to a BIG-IP system for administration purposes. When creating accounts for users to whom you want to grant full privileges, you can choose one of three different roles. The role you choose for a user depends on the type of interface that the user will use to administer the BIG-IP system. Because each role has full access to the BIG-IP system, users with these user roles have privileges to change their own roles or other users' roles.

The roles for fully-privileged users are:

- **Full Web Read/Write**
  This access level provides the user with full access to all administrative tasks. Users with this access level can access the BIG-IP system through the Configuration utility and iControl, but not through the command line interface.
- **CLI + Full Web Read/Write**
  This access level provides the user with full access to all administrative tasks. Users with this access level can access the BIG-IP system through all external interfaces--the Configuration utility, the command line interface, and the iControl interface.

- **CLI**
  This access level provides the user with full access to all administrative tasks, using the command line interface.

Important

The three roles for fully-privileged users all grant the same level of user access, that is, full access to the BIG-IP system. Thus, these roles are not intended as a way to restrict administative access; rather, they are provided strictly as a way to define the method of user access, for administrative convenience.

Restricted users

Restricted users are those whose administrative access to a BIG-IP system is limited. When creating accounts for users to which you want to restrict access, you can choose one of three different roles, where each role represents a different amount of access to the BIG-IP system. The role you choose depends on the amount of restricted access that you want to grant to the user. The roles for restricted users are:

- **Partial Web Read/Write**
  This access level allows the user to view information and to change the status of node addresses to either **enabled** or **disabled**. Users with this access level can access the BIG-IP system through the Configuration utility only.
- **Web Read Only**
  This access level allows the user to view information, using the Configuration utility only. Users with this access level do not have access to **Add** buttons, certain tab items, **Apply** buttons, or **Remove** buttons.
- **None**
  This access level is the default access level, and prevents the user from accessing the BIG-IP system altogether.

The procedure that you use to create and manage user accounts depends on whether you have configured user authentication to use either the local LDAP database that resides on the BIG-IP system, or an external (remote) server. The following sections describe how to assign access levels based on these two different authentication scenarios.

Note

*The **root**, **admin**, and **support** accounts require special consideration when managing them. For information on managing these accounts, see* ***Managing system accounts**, on page 17-29.*

Creating and authorizing local user accounts

When you are using the local LDAP database on the BIG-IP system to authenticate users, your BIG-IP administrative accounts (including user names and passwords) are created and stored in the local LDAP database on the BIG-IP system, using the Configuration utility. Then, you use the Configuration utility to assign a level of access, or user role, to each user account, and upon user authentication, the BIG-IP system checks the local LDAP database to determine the access level that is assigned to that user. An exception to this is the **root** account, which is stored in the UNIX **/etc/passwd** file, rather than in the local LDAP database.

You assign access levels to users at the time that you create their user accounts, or by changing the properties of an existing account.

Creating, changing, and deleting user accounts

You can use the Configuration utility to create new user accounts on the BIG-IP system. For each user account that you create, you can assign one level of access control.

To display a list of existing user accounts using the Configuration utility

1. In the navigation pane, click **System Admin**.
2. Click the User Administration tab.
   This displays a list of all existing local user accounts.

◆ Note

*The Configuration utility only displays those accounts that are stored in the local LDAP database. Thus, the **root** account does not appear in the list of user accounts, given that the account is stored elsewhere.*

To create a user account using the Configuration utility

1. In the navigation pane, click **System Admin**.
2. Click the User Administration tab.
   This displays a list of all local user accounts, except for the **root** account.
3. Click the **Add** button.
4. 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.

5. Choose an access level for the user.
6. Click **Done**.

To change the properties of a user account using the Configuration utility

1. In the navigation pane, click **System Admin**.
2. Click the User Administration tab.
   This displays a list of all local user accounts, except for the **root** account.
3. Click a user account name.
   This displays the properties of that account.
4. Change the password, or choose a new access level for the account.
5. Click **Apply**.

*If you have a redundant system configuration and you change the password on the **admin** account, you must also change the password on the redundant system, to ensure that the **bigpipe config sync** command operates correctly.*

To delete a user account using the Configuration utility

1. In the navigation pane, click **System Admin**.
2. Click the User Administration tab.
   This lists the user roles currently assigned to remote user accounts.
3. In the **Local Users** box, locate a user name for which you want to delete a user role and click the Delete button (trashcan icon). Note that you cannot delete the **admin** user account.

Creating and authorizing remote user accounts

When you are using a remote LDAP or RADIUS authentication server, you create and store your BIG-IP administrative accounts (including user names and passwords) on that remote server, using the mechanism supplied by that server's vendor.

To configure user authorization in this case, you use the Configuration utility and assign a specific access level, or user role, to each remote user account. This access information is then stored in the BIG-IP system's local LDAP database. When a user whose account information is stored remotely

logs into the BIG-IP system and is granted authentication, the BIG-IP system then checks its local LDAP database to determine the access level that is assigned to that user.

If no user role is assigned to a remote user account, then the BIG-IP system assigns access based on a role called the ***Default Role***. Using the Configuration utility, you can set the access level for the Default Role.

The following sections describe the procedures for assigning user roles to remote user accounts.

To display a list of user roles for remote accounts using the Configuration utility

1. In the navigation pane, click **System Admin**.
2. Click the User Administration tab.
   This displays the **Remote User Roles** box, which lists the remote user accounts to which you have assigned an access level, as well as the Default Role and its access level. Also displayed is the **Local Users** box, showing the **admin** account, which is always stored locally on the BIG-IP system.

Important

Any user account that has not been assigned a remote user role automatically inherits the access level assigned to the Default Role.

To assign a user role for a remote account using the Configuration utility

1. In the navigation pane, click **System Admin**.
2. Click the User Administration tab.
3. Click the **Add User Role** button.
4. In the **User ID** box, type a user name that is stored on your remote authentication server.
5. In the **Access Level** box, choose an access level to assign to that user.
6. Click **Done**.

To change a user role for a remote account using the Configuration utility

1. In the navigation pane, click **System Admin**.
2. Click the User Administration tab.
   This lists the user roles currently assigned to remote user accounts.
3. In the **Remote User Roles** box, click a user name.
   This displays the user role properties for that user account.

4. In the **Access Level** box, select a different access level.
5. Click **Apply**.

To delete a user role for a remote accounts using the Configuration utility

1. In the navigation pane, click **System Admin**.
2. Click the User Administration tab.
   This lists the user roles currently assigned to remote user accounts.
3. In the **Remote User Roles** box, locate a user name for which you want to delete a user role and click the Delete button (trashcan icon).

Managing passwords for local user accounts

Sometimes, the users who have accounts stored in the local LDAP database might need to change their passwords. Users can change their passwords by accessing the User Administration screen of the Configuration utility, and then displaying the properties of their user accounts.

This method of changing a password applies not only to the user accounts you create from within the Configuration utility, but also to the **admin** and **support** accounts that the Setup utility created when you configured your base network.

For the procedure on changing passwords for locally-stored user accounts, see *To change the properties of a user account using the Configuration utility*, on page 17-27.

◆ Note

*To change the password for the **root** account, you must re-run the Setup utility. For more information, see the following section.*

Managing system accounts

As previously described, the Setup utility automatically creates three system accounts--**root**, **admin**, and **support**. Only the support account is optional.

These accounts must be managed in the following ways:

- **The root account**
  The **root** account is defined in the **/etc/passwd** file on the BIG-IP system, and therefore does not reside in either the local LDAP database or a remote LDAP database. To initially create the **root** account and set its password, you run the Setup utility. To change its password later, you must re-run the Setup utility. Because the **root** account does not reside in the local or a remote LDAP database, it does not appear on the User Administration screens of the Configuration utility. The access level for this account is fixed during creation and cannot be changed.

- **The admin account**
  The **admin** account is defined in the local LDAP database on the BIG-IP system. To initially create the **admin** account and set its password, you run the Setup utility. To change its password later, you use the Configuration utility's User Administration screens. Note, however, that due to redundant system considerations, you must change the password on both systems of the redundant system configuration, and you cannot delete the password for this account. The access level for this account is fixed during creation and cannot be changed, except when performing an upgrade.
- **The support account**
  The **support** account is defined in the local LDAP database on the BIG-IP system. To initially create the **support** account and set its password, you run the Setup utility. Unlike the **root** and **admin** accounts, however, creation of the **support** account is optional. To change the password and access level for this account later, you use the Configuration utility's User Administration screens.

Working with the bigdb database

The bigdb™ database holds certain configuration information for the BIG-IP system. Most BIG-IP system utilities currently use the configuration stored in bigdb. The **bigpipe db** is provided for loading configuration information into bigdb. An additional **default.txt** file is included with the BIG-IP system which contains default information you can load into the bigdb 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 bigdb configuration key

To display the value of a **bigdb** 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 bigdb configuration key

To create (set) a **bigdb** configuration key, use the following syntax:

```
b db set <key>

To set a bigdb 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 bigdb configuration key

To unset the **bigdb** 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 system.

◆ Note

*The values in the **default.txt** file are default values; several of the keys listed are not present in the bigdb database.*

The **default.txt** file is intended to serve as documentation only. In order for the system to work, 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 bigdb 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 bigdb, see Appendix B, *bigdb Configuration Keys*.

18

Configuring SNMP

- Introducing SNMP administration
- Downloading the MIBs
- Configuring SNMP using the Configuration utility
- SNMP configuration files
- Configuring snmpd to send responses out of different ports or addresses

Introducing SNMP administration

This chapter describes management and configuration tasks for version 3.0 of the simple network management protocol (SNMP) agent. The chapter also describes tasks for the management information bases (MIBs) available with the BIG-IP system.

With the BIG-IP system SNMP agent and MIBs, you can manage the BIG-IP system by configuring traps for the SNMP agent or polling the BIG-IP system with your standard network management station (NMS).

You can use the Configuration utility to configure the BIG-IP system 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 system 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 system 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 system 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 system 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 system 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 18.1. To access this screen, simply click **System Admin** in the navigation pane, then click the SNMP Administration tab.

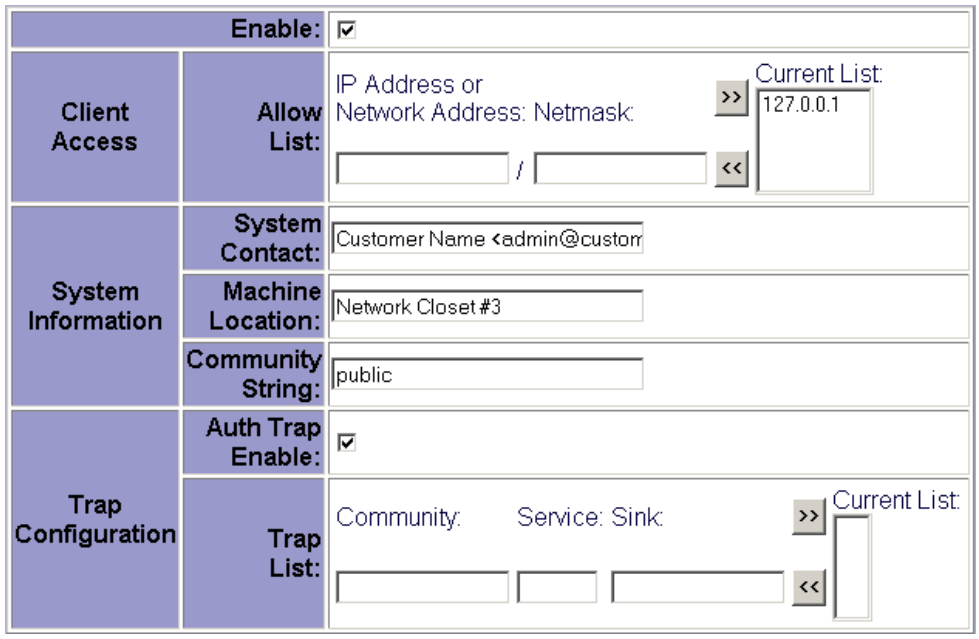

Figure 18.1 *SNMP Administration screen*

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 system 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 system 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 system.
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 system.
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 system 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 system sends traps. Traps sent from the BIG-IP system 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 system 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 system, 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 preceding example 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.

The /etc/snmpd.conf file

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 18.2.

```
# 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 18.2** Excerpt from the **/etc/snmptrap.conf** file*

Some of the OIDs have been permanently mapped to BIG-IP system specific events. The OIDs that are permanently mapped for the BIG-IP system 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).*

A

bigpipe Command Reference

bigpipe commands

This chapter lists the various **bigpipe** commands, including syntax requirements and functional descriptions. Table A.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 A.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.	A-3
authz	Defines a client authorization model for the SSL proxy.	A-4
class	Displays all classes included with BIG-IP system.	A-5
config	Synchronizes the **/config/bigip.conf** between the two BIG-IP units in a redundant system.	A-6
conn	Shows information about current connections such as the source IP address, virtual server and port, and node.	A-8
default_gateway	Creates a pool of default gateways.	A-11
failover	Sets the BIG-IP system as active or standby.	A-12
global	Sets global variable definitions.	A-14
-h and help	Displays online help for **bigpipe** command syntax.	A-25
interface	Sets options on individual interfaces.	A-26

Command	Description	Page
verify	Parses the command line and checks syntax without executing the specified command.	A-74
version	Displays the **bigpipe** utility version number.	A-75
virtual	Defines virtual servers, virtual server mappings, and virtual server properties.	A-76
vlan	Defines VLANs, VLAN mappings, and VLAN properties.	A-79
vlangroup	Defines VLAN groups.	A-81

-?

```
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 -?
```

authz

```
b authz <name> [method (ldap|...)]
    [cachetimeout <number>]
    [ldap searchtype (user|certmap|cert)]
    [ldap servers <server list>]
    [ldap secure (enable | disable)]
    [ldap admindn <string>]
    [ldap adminpw <string>]
    [ldap user base <string>]
    [ldap user key <string>]
    [ldap certmap base <string>]
    [ldap certmap key <string>]
    [ldap certmap useserial <string>]
    [ldap group base <string>]
    [ldap group key <string>]
    [ldap group member key <string>]
    [ldap valid groups <string list>]
    [ldap role key <string>]
    [ldap valid roles <string list>]
b authz <name> method [show]
    b authz <name> cachetimeout [show]
    b authz <name> ldap searchtype [show]
    b authz <name> ldap servers [show]
    b authz <name> ldap secure [show]
    b authz <name> ldap admindn [show]
    b authz <name> ldap adminpw [show]
    b authz <name> ldap user base [show]
    b authz <name> ldap user key [show]
    b authz <name> ldap certmap base [show]
    b authz <name> ldap certmap key [show]
    b authz <name> ldap certmap useserial [show]
    b authz <name> ldap group base [show]
    b authz <name> ldap group key [show]
    b authz <name> ldap group member key [show]
    b authz <name> ldap valid groups [show]
    b authz <name> ldap role key [show]
    b authz <name> ldap valid roles [show]
    b authz <name> delete
    b authz <name> list
    b authz <name> [show]
<server list> ::= <server> <server> ...
    <server> ::= <ip addr>[:<service>] | <host name>[:<service>]
```

Creates or displays the parameter values that make up an SSL proxy authorization model.

Options

The **<name>** variable represents the name of the authorization model being created.

The **[method]** parameter specifies the type of authorization model. Currently, the only valid value for this parameter is **ldap**.

The **[cachetimeout <number>]** parameter specifies the length of timeout for the cache.

The **[ldap searchtype]** parameter specifies the type of search that the proxy will do on the LDAP database in its attempt to authorize a client.

The **[ldap servers <server list>]** parameter specifies the LDAP servers being used for authorization.

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 <class name> member [ add | delete ] { <value list> }
b class <class name> show
b class <class name> member 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 system 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 { \".abc\" ... } | '{ ".def" }'
b class numeric_class { 0 1 ... }
b class host_class { host 1.2.3.0 }
b class network_class { network 1.2.3.0 mask 255.255.255.0 }
```

Options

The **<class name>** variable specifies the name of a class. When creating a class, arguments to the **<class name>** variable can either be **<ip member>**, **<string>**, or **<num>**.

The **<ip member>** variable specifies the IP address of a member that is to be included in the specified class. The **<ip member>** option can be in the form of either **host <ip addr>** or **network <ip addr> mask <ip addr>**.

The **<string>** variable specifies one or more strings to be included in the specified class.

The **<num>** variable specifies one or more numeric values to be included in the specified class.

The **<class name> show** option displays the specified class.

The **<value list>** option lists the members that you want to add to a class.

The **ip show** option displays all classes that contain IP address members.

The **string show** option displays all classes that contain string values.

The **value show** option displays all classes that contain numeric values.

The **show** option with no other arguments displays all classes.

The **<class name> delete** option deletes the specified class.

config

```
b config save <file>
b config install <file>
b config sync
b config sync all
b config sync running
```

Manages user configuration sets. A user configuration set (UCS) is the set of all configuration files that a user may edit to configure a BIG-IP system. A UCS file is an archive that contains all the configuration files in a UCS.

The config command allows you to save the system configuration to a UCS file, install the configuration from a UCS file, and synchronize the configuration with the other systems in a redundant system.

Options

The **config save <file>** option saves the currently running configuration to **/config/bigip.conf** and **/config/bigip_base.conf**, and creates the UCS file with the file name specified by **<file>**.

The **config install <file>** option unpacks and installs the UCS file specified by **<file>**, overwriting all configuration files, including the file **/config/bigip.conf**.

The **config sync** option saves the currently running configuration to the file **/config/bigip.conf** and copies the file **/config/bigip.conf** to the other BIG-IP unit in a redundant system.

The **config sync all** option creates a temporary UCS file and transfers it to the other BIG-IP unit.

The **config sync running** option saves the currently-running configuration to a temporary file and copies it to the other BIG-IP unit.

Saving configuration files to an archive

The config save <file> command 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.*

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 **bigdb** 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 the BIG-IP system and not to 3-DNS.*

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**.

conn

```
b conn [ <client_ip>[:<client_service>] ] dump [verbose]
b conn all dump
b conn all dump delete
b conn dump [verbose] mirror
b conn [ <client_ip>[:<client_service>] ] delete
```

Displays information about current client connections to virtual addresses and virtual servers. This command can also show connections that are active on the given BIG-IP system, as well as those that are standby connections for the peer BIG-IP system. By default, the **dump** command only shows items that are active on the given unit.

Options

The **<client_ip>[:<client_service>]** option specifies the IP address and the service of the system for which you want to display information.

The **all** option indicates that the action applies to all connections.

The **verbose** option displays all current connection information in verbose mode.

The **mirror** option displays information on standby connections.

The **delete** option searches for conections in which the specified IP address and service match the client-side source, and then deletes the connection.

Displaying all current connections

The following command displays all current client connections:

```
b conn dump
```

or

```
b conn all dump
```

The following output shows the source IP address, virtual server IP address, and node to which the client is connected.

```
bigip conn dump

fromvirtual node
100.100.100.30:49152 ->100.100.100.100:23 ->200.200.200.10:23
100.100.101.90:49153 ->100.100.100.100:80 ->200.200.200.10:80
...
```

Figure A.1 *Formatted output of the* ***conn*** *command*

This command can also show connections that are active on the given BIG-IP system, as well as those that are standby connections for the peer BIG-IP system. By default, the **dump** command only shows items that are active on the given unit.

Using verbose mode

The following command displays all current client connections, in verbose mode:

```
b conn dump verbose
```

The following shows sample output from this command:.

```
client side client address: 10.253.220.2:3889
client side server address: 10.253.220.80:80
server side client address: 10.253.220.2:3889
server side server address: 10.253.100.100:80
virtual address: 10.253.220.80:0
node box address: 10.253.100.100:80
protocol: tcp
bytes in: 360
bytes out: 543
packets in: 5
packets out: 5
```

Figure A.2 *Formatted output of the* ***conn verbose*** *command*

Displaying connections for a specific virtual server

Use the following syntax to display the current connections for a specific virtual server:

```
b conn <client_ip>[:<client_service>] dump
```

Note that the argument **<client_service>** refers to what is typically a five-digit number displayed in the output of this command.

Displaying standby connections

To view standby items, you must use the **mirror** qualifier, as follows.

```
b conn dump mirror
```

or

```
b conn dump verbose mirror
```

Deleting connections

To delete all current connections, you use the delete option as follows:

```
b conn delete
```

or

```
b conn all delete
```

When the **delete** option is specified, the **b conn** command searches for conections in which the specified address and service match the client-side source and then deletes the connection. The **b conn** command deletes the connection by sending a **reset** to both the client and the server.

If no service is specified, all connections in which the client-side source matches the IP address are deleted. If no IP address is specified, all connections are deleted.

The **delete** option does not differentiate between mirrored and normal connections. Thus, the option deletes any connections that were mirrored from an active unit.

◆ Note

The delete option does not delete connections managed by the BIG-IP system hardware.

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.

Options

The **use pool <pool_name>** option specifies the name of the default gateway pool and must be 1-31 characters in length. Example: **my_pool**.

The **show** option shows the members of the default gateway pool.

The **delete** option deletes the default gateway pool.

failover

```
b failover standby | show | init | failback | linkdown <seconds>
```

Switches the BIG-IP system or 3-DNS controller to be the standby unit in a redundant configuration. This command should be used with care, and is provided only for special situations. The BIG-IP system or 3-DNS Controller automatically switches between active and standby modes, without operator intervention.

Options

The **standby** option switches the BIG-IP system to the standby unit.

The **show** option displays the node on which the BIG-IP system is currently running.

The **init** option initializes the initial state of the BIG-IP system.

The **failback** option restorse an active-active configuration after a failure. This option is only valid when the BIG/store key **Common.Bigip.Failover.ManFailBack** has been created and set to a value of **1**.

The **linkdown <seconds>** option specifies the amount of time to bring the interfaces down when the unit fails over to the standby unit. Used to prompt peer switch appliances into resetting and relearning their ARL tables after a failover.

Changing failover state

Before you switch the current mode, first determine which mode the BIG-IP system or 3-DNS Controller is running using the command above. In an active/standby or active-active configuration, run the following command to switch the system to be the standby unit:

```
b failover standby
```

Displaying failover status

Show the status of the BIG-IP system or 3-DNS Controller with the following command:

```
b failover show
```

Initializing failover state

You can use the **bigpipe failover init** command to refresh the parameters of the failover mechanism with any new configuration data entered into the **bigdb** database.

```
b failover init
```

Restoring an active-active configuration after failure

This command will only work when the BIG/store key **Common.Bigip.Failover.ManFailBack** has been created and set to a value of one.

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.*

Specifies the length of time that a BIG-IP unit in a VLAN group should keep its links down when they are dropped during a switch from active to standby mode. The value is specified in tenths of seconds. Thus, a value of **50** is equivalent to 5 seconds. By default, this feature is disabled, with a value of **0**. The following command specifies a value of 5 seconds:

```
b global set standby_link_down_time = 50
```

global

```
b global audit enable | disable | verbose | show
b global auto_lasthop enable | disable | show
b global broadcast [ accept | discard ]
b global fastest_max_idle_time <seconds>
b global fastflow_active auto | on | off | show
b global gateway failsafe arm | disarm | show
b global idle_http_conn_timeout <num_val>
b global idle_http_conn_timeout show
b global ipforwarding enable | disable
b global memory_reboot_percent <percent>
b global mirror enable | disable | show
b global mirror_vlan_forwarding enable | disable | show
b global msrdp no_session_dir enable | disable
b global open_3dns_ports enable | disable | show
b global open_corba_ports enable | disable | show
b global open_failover_ports enable | disable | show
b global open_ftp_ports enable | disable
b global open_radius_ports enable | disable | show
b global open_rsh_ports enable | disable
b global open_snmp_ports enable | disable | show
b global open_ssh_port enable | disable
b global open_telnet_port enable | disable
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 reaper lowater <percent>
b global reaper hiwater <percent>
b global self_conn_timeout enable | disable | show
b global snats any_ip enable | disable
b global sslproxy akamaizer filename <filename>
b global sslproxy akamaizer service <service>
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 vlangroups opaque | translucent | transparent
b global vlans lookup enable | disable
b global vlans unique_mac enable | disable
b global webadmin_port <port>
b global web aggregate all | ip | port | none
b global web aggregate timeout <seconds>
b global web escapes decode | ignore
b global web parse first | all
b global l2_aging_time <seconds>
```

audit

This variable logs all create, delete, and modify actions on the BIG-IP system.

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**.

broadcasts

This variable controls the acceptance or rejection of IP broadcast packets by the BIG-IP system.

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 system to send fewer connections to a node that is responding slowly, and also allows the BIG-IP system 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
```

ip forwarding

Enables IP forwarding for the BIG-IP system. IP forwarding exposes all of the node IP addresses to the external network, making them routable on that network. The default setting is **disabled**.

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.

memory_reboot_percent

The value you type, **80** or higher, is the percentage of memory that is in use before the BIG-IP system automatically reboots. The default value for this variable is **97**. To disable this feature, set the value to **0**.

mirror

Enables mirroring functions globally for the BIG-IP system. 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**.

mirror_vlan_forwarding

This variable is used to forward packets from a mirror-target VLAN to a source VLAN, after an intrustion detection system has attempted to terminate a connection.

msrdp no_session_dir

This variable is used to implement Windows Terminal Server persistence for those Windows servers on which the Session Directory service is not available.

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 system itself. The variable is disabled on the BIG-IP system 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 that allow administrative CORBA connections. The default setting is **disabled**.

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
```

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_radius_ports

This variable is required for RADIUS authentication. Enabling this variable allows the kernel to safely send UDP traffic on external, locked-down ports, without compromising the shared RADIUS secret sent between client and server.

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_snmp_ports

This variable enables and disables the SNMP ports that allow administrative SNMP connections. The default setting is **disabled**.

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_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
```

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**.

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 A.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 A.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 timeout
```

The following command resets the timer only when the persistent connection is **initiated**.

```
b global persist timer limit
```

◆ Note

For SSL persistence, the timer is always reset on each packet.

reaper hiwater

Used to prevent denial-of-service attacks, this variable specifies a high-watermark threshold for determining when unestablished connections through the BIG-IP system will no longer be allowed. The value of this variable represents a percentage of memory utilization. Once memory utilization has reached this mark, connections are disallowed until the available memory has been reduced to the low-watermark threshold. For example, the following command specifies that connections will no be allowed when memory utilization reaches 95%:

```
b global reaper hiwater 95
```

Setting this value to **100** disables the feature. See also *reaper lowater*.

reaper lowater

Used to prevent denial-of-service attacks, this variable specifies a low-watermark threshold for determining at what point adaptive reaping becomes more aggressive. For example:

```
b global reaper lowater 85
```

The default setting for this variable is 85. Setting this value to **100** disables the feature. See also *reaper hiwater*.

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**.

snats any_ip

When this variable is disabled, the BIG-IP system attempts to forward an any-IP packet originating from a member of a SNAT, instead of rejecting that packet.

sslproxy akamaizer filename

This variable specifies the SSL proxy akamaizer filename. The default file name is **/config/akamai.conf**.

sslproxy akamaizer service

This variable specifies the SSL proxy akamaizer service. The default service is **0**, representing no service.

sslproxy failover

This variable causes the SSL proxy to initiate an automatic failover, in the event of a fatal failure of a cryptographic hardware module. Two settings are allowed--**enable** and **disable**. The default setting is **disable**.

Note that this use of this variable depends on the type of hardware module, as not all hardware modules respond to failures in the same way.

sslproxy serverssl cache size

This variable specifies the maximum size of the server-side SSL session cache. The default value is 20,000 entries. A value of **0** disallows session caching. Note that this value is for server-side cache size only. Client-side cache size is configured on a per-proxy basis, using the **bigpipe proxy** command.

sslproxy serverssl cache timeout

This variable specifies a timeout value for the server-side SSl session cache. Note that this value is for the server-side cache timeout only. Client-side cache timeout is configured on a per-proxy basis, using the **bigpipe proxy** command.

sslproxy strict resume

This variable allows the SSL proxy to either resume or not resume the SSL sessions after an unclean shutdown. The two settings are **enable** and **disable**. The default setting is **disable**, which causes the SSL proxy to allow uncleanly shut down SSL sessions to be resumed.

sslproxy unclean shutdown

This variable causes the SSL proxy to perform either a clean or an unclean shutdown of all SSL connections. The default setting is **enable**, which causes the SSL proxy to perform unclean shutdowns. To force the SSL proxy to perform clean shutdowns, you use the **disable** option.

sticky table_limit

This is the maximum number of sticky entries allowed to accumulate on the BIG-IP system when using destination address affinity (sticky persistence). When the maximum value is reached, the BIG-IP system stops accumulating sticky entries. The default value for this entry is **2048**.

verbose_log_level

This variable specifies logging levels for both TCP and UDP traffic. To set this logging level, specify a number. The default setting is **0**, representing no logging. Table shows the result of providing various values on the command line.

b global verbose_log_level command	Result
b global verbose_log_level 1	This option logs attempts by a client to connect to an unauthorized UDP port on the BIG-IP system.
b global verbose_log_level 2	This option logs attempts by a client to connect to an unauthorized TCP port on the BIG-IP system.
b global verbose_log_level 4	This option logs attempts by a client to connect to an unauthorized UDP port on a virtual address.

Table A.3 Possible values for the verbose_log_level global variable

b global verbose_log_level command	Result
b global verbose_log_level 8	This option logs attempts by a client to connect to an unauthorized TCP port on a virtual address.
b global verbose_log_level 15	This option turns on UDP and TCP port denial for both virtual server and BIG-IP addresses.
b global verbose_log_level 16 **b global verbose_log_level 32**	These options log attempts to reset statistics on objects.
b global verbose_log_level 64	This option logs messages regarding FTP connection diagnostics.
b global verbose_log_level 128	This option logs security messages regarding denial of general IP connections.These messages are not specific to TCP or UDP connections.
b global verbose_log_level 256	This option logs messages regarding SSL connection diagnostics.

Table A.3 Possible values for the verbose_log_level global variable

For example, 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
```

vlangroups

Enables layer 2 operation for VLAN groups. By default, VLAN groups are a hybrid of layer 2 proxy ARP with layer 3 forwarding. Using this variable, you can change the way that VLAN groups operate. Available settings for this variable are:

opaque
A proxy ARP with layer 3 forwarding. The command line syntax for enabling this setting is:

```
b global vlangroups opaque
```

translucent
Layer 2 forwarding with locally-unique bit, toggled in ARP response across VLANs. This is the default setting.

transparent
Layer 2 forwarding with the original MAC address of the remote system preserved across VLANs. The command-line syntaxfor enabling this setting is:

```
b global vlangroups transparent
```

vlans lookup

Enables VLAN-keyed connections. VLAN-keyed connections are used when traffic for the same connection must pass through the BIG-IP system several times, on multiple pairs of VLANs (or in differenct VLAN groups). The default setting is **enable**. To disable this feature, use the following command:

```
b global vlans lookup disable
```

vlans unique_mac

This variable is used to circumvent problems caused by problematic switch appliances that do not keep per-VLAN L2 forwarding tables. When set to **enable**, this variable causes VLANs to assume the MAC address of the first non-hidden member interface. When set to **disable**, a single MAC address is used for all VLANs.

The default setting for server appliances is **enable**. The default setting for switch applicances is **disable**.

If a non-default setting is used, you must set the variable before you load any VLANs. Also, you must reload the base configuration after changing the setting of this variable.

webadmin_port

Specifies the port number used for administrative web access. The default port for web administration is port **443**.

web aggregate

This variable provides fine-grained control of client aggregation. Possible settings are:

all
Causes all clients, regardless of IP address, to be piggy-backed on established idle connections to servers.

ip
This is the default setting. Causes all clients with the same IP address to be piggy-backed on established idle connections to servers.

port
Causes only requests from both the same client source IP address and source port to be aggreagated. This behavior is required for enabling non-compliant HTTP implementations to use keep-alives.

none
Establishes a connection to back-end servers with each new request in the front-end stream, regardless of the viability of idle connections.

web aggregate timeout

This variables allows you to configure a timeout value, in seconds, for the idle HTTP connection reaper. The minimim timeout value allowed is **1**. The maximum timeout value is INT_MAX, currently defined as **2147483647**. The default timeout value is **5**.

web escapes

This variable decodes "%" escape characters in URIs before comparison, using a rule. The variable can be set to **decode** or **ignore**. The default setting is **ignore**.

web parse

The **first** option to this variable disables both aggregation and keep-alive parsing, reverting the BIG-IP system to its pre-4.0 behavior. The default setting is **all**.

-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 show
b interface [<interface_name>] show [verbose]
b interface <inteface_name> media show
b interface <inteface_name> duplex show
b interface <interface_name> media <media_type>
b interface <interface_name> duplex full | half | auto
b interface [<interface_name>] stats reset
b interface <interface_name> enable | disable
b interface <interface_name> renames <driver_name>
```

Displays the names of installed network interface cards and, for each interface, sets properties such as MAC address, media options, duplex mode, and status, resets interface statistics, enable or disable interfaces, and change driver name mappings.

Options

The **<interface_name>** variable is a name such as **3.1**, where **3** is the physical slot number holding the network interface hardware and **1** is the physical port number on that interface on that hardware.

The **show [verbose]** option displays the current status, settings, and network statistics for the specified interface. The **verbose** argument provides more detailed information. If no interface is specified, this option displays information for all interfaces.

The **media show** option displays information about the media type for the specified interface.

The **duplex show** option displays the duplex mode of the specified interface.

The **media <media_type>** option is a valid media type for the specified interface. Examples include **auto**, **100baseTX**, and **10baseT**. Note that only certain combinations of media type and duplex mode are valid for any particular type of interface.

The **duplex full | half | auto** option sets the duplex mode of the specified interface.

The **stats reset** option resets the statistics for the specified interface.

The **enable | disable** option enables or disables the specified interface.

The **renames <driver_name>** option changes the mapping from the interface's driver name to its physical location name. The **<driver_name> option** is the network interface name in the form of driver and unit number, such as **exp0** and **bs1**. Note that this is the old-style network interface name.

Displaying interface information

To display the status, settings, and statistics for all interfaces on the BIG-IP system, use the following command.

```
b interface show [verbose]
```

To display the status, settings, and statistics for a specific interface on the BIG-IP system, use the following command-line syntax.

```
b interface <interface_name> show [verbose]
```

Note that if the **verbose** argument is used, the output provides additional information on status. If the **verbose** argument is not used, the output focuses on statistics.

To display the media type for an interface, use the following command-line syntax,

```
b interface <interface_name> media show
```

To display the duplex mode for an interface, use the following command-line syntax.

```
b interface <interface_name> duplex show
```

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**.

To set the media type, use the following command-line syntax.

```
b interface <interface_name> media <media_type>
```

Setting the duplex mode

Duplex mode may be set to **full**, **half duplex, or auto**. 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.

To set the duplex mode, use the following command-line syntax.

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

Resetting statistics

You can reset interface statistics for all interfaces or for a specific interface.

To reset statistics for all interfaces, use the following command.

```
b interface stats reset
```

To reset statistics for a specific interface, use the following command-line syntax:

```
b interface <interface_name> stats reset
```

Enabling or disabling an interface

Enabling or disabling an interface allows you to control whether the interface receives and sends packets. If an interface begins to behave strangely, you disable and then enable the interface to effectively reset it.

To enable or disable an interface, use the following command-line syntax.

```
b interface <interface_name> enable | disable
```

Changing driver name mapping

You can change the mapping from an interface's driver name to its physical location name, using the following syntax.

```
b interface <interface name> renames <driver name>
```

load

```
b [verify] load [ <filename> | - ]
b [-log] load [ <filename> | - ]
```

Resets all of the BIG-IP system 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 system into and out of maintenance mode. When in maintenance mode, a BIG-IP system 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 system is already in maintenance mode, the **maint** command takes the BIG-IP system out of maintenance mode. If the BIG-IP system is in maintenance mode for more than 20 minutes, that BIG-IP system immediately begins to accept new connection requests.

If the BIG-IP system 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 up the process by reloading the configuration file, using the following command:

```
b -f /config/bigip.conf
```

makecookie

```
b makecookie <ip_addr:service>
```

Creates a cookie template similar to the templates shown in Figure A.3 and Figure A.4. The command generates a cookie string with encoding automatically added for the Passive mode of cookie persistence. The comand-line syntax is as follows:

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

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

Figure A.3 *Sample cookie template*

```
Set-Cookie:BIGipServer[poolname]=336268299.20480.0000; expires=Sat, 01-Jan-2000
00:00:00 GMT; path=/
```

Figure A.4 *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>]
```

Loads the BIG-IP system configuration from the file specified in the **<file_name>** variable, without resetting the current configuration.

mirror

```
b mirror [<mirror_to_interface>] show
b mirror <mirror_to_interface> interfaces add <interface_list>
b mirror <mirror_to_interface> interfaces delete <interface_list>
b mirror <mirror_to_interface> delete
```

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.

Options

The **<mirror_to_interface>** variable specifies the port to which you want one or more ports to be mirrored.

The **show** option displays a specific mirror-to interface. If no interface is specified, this option displays all mirror-to interfaces.

The **interfaces add <interface_list>** variable specifies one or more ports that you want to mirror to the mirror-to port.

The **interfaces delete <interface_list>** variable specifies one or more ports that you want to delete from a port mirror.

The **delete** option deletes the specified mirror-to interface.

Displaying port mirroring

Using the argument, you can display all mirror-to interfaces or a specific mirror-to interface.

To display all mirror-to interfaces, type the following command:

```
b mirror show
```

To display a specific mirror-to interface, use the following command-line syntax:

```
b mirror <mirror_to_interface> show
```

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_interface> interfaces add <interface_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

The **bigpipe** syntax for deleting interfaces from a port mirror is as follows:

```
b mirror <mirror_to_interface> interfaces delete <inteface_list>
```

For example, you could type the following command:

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

Deleting a port mirror

The **bigpipe** syntax for deleting a port mirror is:

```
b mirror <mirror_to_interface> 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 <monitor_name> show
b monitor dump [all]
b monitor <name> delete
b monitor <name> enable | disable
b monitor instance <ip_address>:<service>  enable | disable
b monitor instance <ip_address> 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.

Monitors verify services and connections of node servers. The **icmp** or **tcp_echo** monitors may be used to monitor node addresses. If the node server or node address fails to respond in the specified timeout period, it will be markedas **down**. When a node server or node address is marked as **down**, traffic is no longer directed to it.

Several steps are needed to create a monitor and associate it with a node server or node address. A monitor must be created, based on a monitor template that the BIG-IP system provides. In some cases, a monitor template is directly usable. Once a monitor is created, the node address or node server is associated with the monitor, creating a monitor instance.

Options

The **<monitor_name>** variable specifies the name you want to use for the monitor you are creating or managing.

The **<monitor_template>** variable specifies the health monitor template you want to use to create your monitor. For a list of templates that you can specify, see *Monitor templates*, on page A-36.

The **<attr>** variable specifies an attribute of the monitor to which you want to assign a value. For a list of monitor attributes, see *Monitor templates*, on page A-36.

The **<attr_value>** variable specifies the value of the attribute specified with the **<attr>** option.

The **show all** option displays all existing monitors.

The **show** option displays the specified monitor.

The **delete option deletes the specified monitor.**

The **enable | disable** option enables or disables the specified monitor.

The **instance <ip address>:<service>** option enables or disables a monitor instance for the specified IP address and port.

The **instance <ip address>** option enables or disables a monitor instance for the specified IP address.

Creating a monitor

Creating a monitor simply names and sets the options for a monitor, based on a monitor template. The options may be obtained from a predefined set of default options or the option values may be specified on the command line during creation.

Options include destination address, interval time, timeout value, send string, and receive string, etc. Options can be changed later using the **modify** option.

The following is an example of a command to create an **http** monitor:

```
b monitor my_http '{ use http send "GET /my.html" recv "TESTING"
}'
```

The command above creates a monitor with the name **my_http**, based on the **http** template. The **send** and **recv** strings are modified from the default values. The **interval**, **timeout**, **destination address**, **username,** and **passwd** configuration options are not specified on the command line because the monitor will use the default values.

Note that single quotes are used when entering monitor commands on the command line, to prevent the command shell from attempting to interpret the double quotes within the monitor definition.

Modifying a monitor

If you want to change the default values of certain options, such as **interval** and **timeout**, you can use syntax as in the following example:

```
b monitor my_http '{ interval <seconds> timeout <seconds> }'
```

Creating a monitor instance

Creating a monitor instance simply associates a monitor or group of monitors with a node address or node server.

Each monitor template contains a destination address option. Almost always, this is the meta character string "*:*", which causes the BIG-IP system to create the monitor instance using the IP address and port supplied on the command line. For example, the destination address option **dest** in the **tcp** monitor template looks as follows:

```
monitor tcp {

        # type tcp
        interval 5
        timeout 16
        dest *:*
        send ""
         recv ""

}
```

We can create two instances of this monitor by entering the following command:

```
b node 10.10.10.10:80 10.10.10.12:80 monitor use tcp
```

The **dest *:*** attribute in the **tcp** monitor causes the two monitor instances to be created, substituting the IP address and port combination supplied on the command line into the destination address. In other words, there are two monitor instances created, one that communicates with address **10.10.10.10:80**, and one that communicates with **10.10.10.12:80**. The node **10.10.10.10:80** depends on the monitor instance **10.10.10.10:80**. If the monitor instance cannot get a response from node **10.10.10.10:80**, then the node is marked as **down**. The same is true for node **10.10.10.12:80**.

It is also possible to enter explicit addresses into a monitor. For example, the following shows a monitor called exp_tcp that specifies an explicit destination address:

```
monitor exp_tcp {
        # type tcp
        use "tcp"
        interval 5
        timeout 16
        dest 10.10.10.24:80
        send ""
        recv ""
 }
```

In this case, the following command causes one monitor instance to be created, one that communicates with address **10.10.10.24:80**:

```
b node 10.10.10.10:80 10.10.10.12:80 monitor use exp_tcp
```

In this case, the nodes **10.10.10.10:80** and **10.10.10.12:80** depend on the health of node **10.10.10.24:80**. If that node does not respond, both **10.10.10.10:80** and **10.10.10.12:80** are marked as **down**.

The following is another example of specifying a destination address on the command line:

```
b node '*:http' monitor use my_http
```

The command above creates a monitor instance for all node addresses with a service of **http**. Note that it is necessary to enter the single quotes when entering this command on the command line to prevent the shell from interpreting the special character *.

Modifying a monitor instance

The **enable/disable** attribute can be changed within a monitor instance. For example:

```
b monitor instance 10.20.3.2:http disable
```

This command disables a monitor instance for a node server. The monitor will not attempt to establish a connection with the service until it is later enabled.

Deleting a monitor

To delete a monitor, use the **bigpipe monitor** command with the **delete** option, as in the following example:

```
b monitor my_http delete
```

Deleting a monitor instance

To delete a monitor instance, use the **bigpipe node** command with the **delete** option, as in the following example:

```
b node '*:http' monitor delete
```

Displaying monitor templates

To display a specific monitor template, use the following command-line syntax:

```
b monitor <monitor template> show
```

When you issue the above command, the BIG-IP system displays the specified template.

To display all monitor templates, use the following command:

```
b monitor show all
```

Displaying monitor instances

Using the bigpipe node command, you can display the status of a monitor instance, along wiht the corresponding node status. For example:

```
bigpipe node 192.168.200.50:http monitor show
```

To see this information for all monitor instances, use the following command:

```
b node monitor show
```

Monitor templates

Table A.4 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)

Table A.4 *The monitor templates*

Name/Type	Template-Specific Attribute Set
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)
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)

Table A.4 *The monitor templates*

Name/Type	Template-Specific Attribute Set
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 A.4 *The monitor templates*

Table A.5 on the next page defines the attributes used in the monitor 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
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 A.5 *Monitor attributes*

-n

```
b -n
```

Used 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 A.5 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 A.5 *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 a network addtress translation (NAT), which is 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** command command defines a mapping between the IP address of a server behind the BIG-IP system **<orig_addr>** and an unused routable address on the network in front of the BIG-IP system **<trans_addr>**.

The primary reason to define a NAT is to allow one of the servers in the server array behind the BIG-IP system to initiate communication with a computer in front of or external to the BIG-IP system.

Options

The **<orig addr>** variable is the originating IP address.

The **<trans addr> variable** is the translated IP address.

The **unit <unit ID>** option specifies a unit ID, currently **1** or **2**. The default unit ID is set to **1**.

The **delete option** deletes a NAT from the BIG-IP system.

The **stats reset** option resets statistics for the specified NAT.

The **vlans <vlan_list>** option lists the existing VLANs on which access to the NAT is enabled or disabled. A NAT is accessible on all VLANs by default.

The **vlans delete all** option deletes the specified NAT for all VLANs.

The **vlans show** option displays the VLANs on which the specified NAT is enabled.

Defining a NAT

Use the following syntax to define a NAT:

```
bigpipe nat <orig ip> to <trans ip> [unit <id>] [arp disable] \
[vlans <vlan name>... disable]
```

The node behind the BIG-IP system with the IP address specified by **<orig ip>** has a presence in front of the BIG-IP system as IP address **<trans ip>**.

For example:

```
b nat 11.0.0.100 to 10.0.140.100
```

Deleting a NAT

Use either of the following commands to permanently delete one or more NATs from the BIG-IP system configuration:

```
b nat <orig_addr>... <orig_addr> delete
```

```
b nat <trans_addr>... <trans_addr> delete
```

Additional Restrictions

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

- A virtual server cannot use the IP address defined in the **<trans ip>** parameter.
- A NAT cannot use the IP address of a BIG-IP system.
- The IP address defined in the **<orig ip>** parameter must be routable to a specific server behind the BIG-IP system.
- A NAT cannot use an orignating or translated IP address defined fo and used by a SNAT or another NAT.
- You must delete a NAT before you can redefine it.

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.**

Options

The **<node_ip>[:<service>]** variable is an IP address of the node address.

The **enable | disable** options enable or disable traffic for one or more specified IP addresses.

The **limit <max_conn>** option defines the maximum number of connections allowed for one or more specified nodes.

The **stats reset** option resets statistics for the specified node.

The **up | down** option causes a node to change to the "forced up" or "forced down" state.

The **monitor use <monitor_name>** option associates one or more specified monitors with the specified node.

The **monitor show | delete** option shows or deletes a monitor instance running on the specified node.

Displaying nodes

You can display information about a specified node. For example, the following command displays information about node `192.168.200.50:20`:

```
b node 192.168.200.50:20 show
```

Note that the **show** keyword is optional.

The resulting information displayed is as follows:

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

Modifying nodes

Use the following syntax to set the maximum number of connections allowed for one or more nodes:

```
b node <ip addr>:<port>... <ip addr>:<port> limit <limit>
```

Note that to remove a connection limit, you also issue the above command, but you set the **<limit>** variable to **zero**.

Use the following syntax to set the maximum number of connections allowed for one or more IP addresses:

```
b node <ip addr>... <ip addr> limit <limit>
```

Note that to remove a connection limit, you also issue the above command, but you set the **<limit>** variable to **zero**.

Use the following syntax to enable or disable traffic for one or more IP addresses:

```
b node <ip addr>... <ip addr> enable
b node <ip addr>... <ip addr> disable
```

◆ Note

*For information on using the **bigpipe node** command to associate a node with a health monitor, see **monitor**, on page A-32 .*

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> header erase <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 simple [simple_timeout <timeout>] | cookie [cookie_expire
    <timeout>] | ssl [ssl_timeout <timeout>] | sip [sip_timeout <timeout>] | sticky
    [sticky_timeout <timeout>] | msrdp [msrdp_timeout <timeout>] | (<expression>)
    [persist_timeout <timeout>] }
b pool <pool_name> select (<expression>)
b pool <pool_name> sticky clear
b pool <pool_name> stats reset
```

Displays, creates, modifies, or deletes a pool definition. You can use pools to group members together with a common load-balancing mode and persistence mode.

Options

The **<pool name>** variable is a string from 1 to 31 characters, for example, **new_pools**.

The **<member_definition>** variable specifies the IP address of the member node being added to the pool.

The **<cookie name>** variable specifies a cookie name, which must be 1-31 characters in length.

The **lb_method <lb_method_specification>** option specifies the load balancing mode that the BIG-IP system is to use for the specified pool.

The **persist <persist_mode_specification>** option specifies the persistence type that BIG-IP system is to use for the specified pool.

The **select <expression>** option specifies the data to be used to directly select a node in the pool.

The **min_active_members <min_value>** option specifies the minimum number of members that must remain available for traffic to be confined to a priority group when using priority-based activation.

The **fallback** option specifies HTTP redirection, using a set of format strings. You can use these strings to indicate unchanged host names, ports, and URI paths. For more information, see *Specifying HTTP redirection*, on page A-46.

The **forward** option specifies that the pool is to be a forwarding pool.

The **snat disable** option specifies that SNAT connections are to be disabled for that pool.

The **mirror** option mirrors a persistence record over to a standby unit. The persistence record identifies the connections to be persisted.

Displaying a pool

Using the **bigpipe pool** command, you can display specific pools or all pools, and display persistence within a pool.

Use the following syntax to display all pools:

```
bigpipe pool show
```

Use the following syntax to display a specific pool, such as **cgi_pool**:

```
bigpipe pool cgi_pool show
```

Use a c ommand such as the following to display persistence within a pool:

```
bigpipe pool cgi_pool persist show
```

Creating a pool

To create a pool, use command-line syntax such as the following:

```
bigpipe pool cgi_pool {  lb_method rr  member 10.2.3.11:http \
member 10.2.3.12:http  }
```

This command creates a pool with two members **10.2.3.11** and **10.2.3.12,** and both members use the **round robin** load balancing method.

If the **lb_method** option is not set, it defaults to round robin.

To create a pool using simple persistence, use command-line syntax such as the following:

```
bigpipe pool cgi_pool {  lb_method rr  persist_mode simple \
simple_timeout 100  simple_mask 255.255.255.0 \
member 10.20.3.11:http member 10.20.3.12:http  }
```

This command creates a pool with two members, **10.20.3.11** and **10.20.3.12**.

Both members use the **round robin** load balance method. Also, a simple persistence timeout of 100 seconds will be used with this pool. Note that an optional persistence mask may be specified with simple persistence.

Modifying a pool

You can modify a pool to change the defined attributes, such as adding or deleting members, changing the load balancing method, or changing the type of persistence being used.

The following example adds a new member to the existing pool **cgi_pool**:

```
bigpipe pool cgi_pool add { member 10.20.3.2:http }
```

The following example deletes a member from the existing pool **cgi_pool**:

```
bigpipe pool cgi_pool delete { member 10.20.3.2:http }
```

Deleting a pool

You can delete a pool altogether. For example, the following command deletes the pool **cgi_pool**:

```
bigpipe pool cgi_pool delete
```

Note that all references to a pool must be removed before a pool can be deleted.

Specifying HTTP redirection

To specify HTTP redirection (also known as fallback), you can use a set of format strings to indicate unchanged host names, ports, and URI paths. These format strings are as follows:

- **%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

For example, the following command configures a pool to redirect an HTTP request from **http://www.siterequest.com:8080/sample.html** to **https://www.siterequest.com:443/sample.html**:

```
bigpipe pool my_pool fallback https://%h:443/%u
```

To indicate that the host name, port, and URI path remain unchanged, you would use the following command:

```
bigpipe pool my_pool fallback %h:%p/%u
```

Specifying a load balancing mode

The load balancing modes are specified as values of the attribute **lb_mode**. The **lb_mode** values are shown in Table A.6.

Mode Name	lb_mode attribute value
Round Robin	rr or omit **lb_mode** specification
Ratio	ratio
Ratio Member	ratio_member
Fastest	fastest
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 A.6 *Load balancing modes*

For more information about the load balancing modes, see Chapter 4, *Pools*.

power

```
b power [show]
```

Allows the user to query for the status of two power supplies in a redundant power supply configuration. Failover from one power supply to the other occurs transparently to the user.

Options

The **[show]** option displays the status of the two power supplies, as follows:

```
b power [show]
  top power supply: active
  bottom power supply: down!
```

proxy

```
b proxy <ip>:<service> [unit <id>][{] target <virtual|server>> <ip>:<service>
    [clientssl] <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] authenticate depth <num>]
    [[clientssl] crl file <clientside CRL file name>]
    [[clientssl] crl path <clientside CRL path>]

    [serverssl crl file <serverside CRL file name>]
    [serverssl crl path <serverside CRL path>]
    [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]
    [tcp connlimit <limit>]
    [}]
b proxy <id addr>:<service> authz set auth hdr (enable | disable)]
    b proxy <id addr>:<service> authz set remoteuser hdr (enable | disable)]
    b proxy <id addr>:<service> authz models <models list>]
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 show
    b 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
```

proxy

```
    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
    b proxy [<ip:service>...] tcp connlimit show

b proxy <id addr>:<service> authz set auth hdr [show]
    b proxy <id addr>:<service> authz set remoteuser hdr [show]
    b proxy <id addr>:<service> authz models [show]
```

Creates, deletes, modifies, or displays the SSL or content converter proxy definitions on the BIG-IP system. For detailed information about setting up the SSL Accelerator feature, see the *BIG-IP Solutions Guide, Chapter 11, Configuring an SSL Accelerator*. For detailed information about setting up the content converter feature, see the *BIG-IP Solutions Guide, Chapter 16, Configuring a Content Converter.*

Options

The **<clientssl> <enable | disable>** option enables and disables the client-side SSL connection feature for the proxy being defined. If this option is omitted, the default is to have SSL **enabled** for all client-side connections.

The **<clientssl key> <clientside key file name>** option specifies a key file to be used as the private key corresponding to the clientside cert file specified by the **<clientssl cert> option**. This option is required when clientside SSL is enabled.

The **<clientssl cert> <clientside cert file name>** option specifies a certificate file to be used as the public key corresponding to the client-side key file specified by the **<clientssl key>** option. This certificate is used as a server certificate when the proxy authenticates itself to clients. This option is required when client-side SSL is enabled.

The **<clientssl chain> <clientside chain file name>** option specifies a chain file to be used to complete the certificate chain corresponding to the clientside cert file specified by the **<clientssl cert>** option. Certificates from this file are used as necessary to build up the certificate chain. This option is not required, even when client-side SSL is enabled.

The **<clientssl ca file> <clientside CA file name>** option specifies a CA file to be used primarily to authenticate client certificates, but also to complete the certificate chain corresponding to the client-side certificate file specified by the **<clientssl cert>** option. See also the **<clientssl chain>** option. The CA file may contain more than one certificate. This option is not required, even when client-side SSL is enabled.

The **<clientssl ca path> <clientside CA path>** option specifies a path to a directory with certificate files to be used primarily to authenticate client certificates, but also to complete the certificate chain corresponding to the clientside cert file specified by the **<clientssl cert>** option. See also the **<clientssl chain>** and **<clientssl ca file>** options. Unlike the **<clientssl ca file>** option, only the first certificate in certificate files with valid symbolic links generated by the **<make>** option will be examined (the Makefile in the **/config/bigconfig/ssl.crt/** directory should be used). This option is not required, even when client-side SSL is enabled.

The **<clientssl client cert ca> <clientside client cert CA file name>** option specifies a CA file containing one or more certificates to be advertised to clients as those CAs trusted for client authentication. Note that this list of CAs has no effect on and can be completely different from those actually used to authenticate clients; see the **<clientssl ca file>** option. If this option is not specified, no list of trusted CAs will be advertised to clients, which may prevent some clients from connecting when client certificates are requested or required.

The **<clientssl cipher insert> [<enable | disable>]** option enables and disables the prepending of an HTTP header containing the negotiated client cipher information. This header takes the form of **"SSLClientCipher: <cipher>, version=<SSL-protocol-version>, bits=<cipher-strength-bits>"**. If this option is omitted, the default is to have client cipher insertion disabled.

The options **<clientssl client cert insert> ([versionnum] [serial] [sigalg] [issuer] [validity] [subject] [subpubkey] [x509ext] [whole] [hash])+** enable the prepending of HTTP headers containing the client certificate information.

The **<clientssl sessionid insert> ([initial] [current])+** option enables the prepending of HTTP headers containing the initial and/or current SSL session ID. These headers take the form of **"SSLClientSessionID: <InitialSessionID>"** and **"SSLClientCurrentSessionID: <CurrentSessionID>"** respectively, where the **<InitialSessionID>** and

<CurrentSessionID> options are the hexadecimal representation of the corresponding SSL session ID. If this option is omitted, the default is to have client session ID insertion disabled.

The **<clientssl ciphers> <list>** option uses the **<list>** option to determine the set of ciphers available for client-side SSL negotiation. See **<http://www.openssl.org/docs/apps/ciphers.html>** for the format of **<list>**.

The **<clientssl invalid> [SSLv2] [SSLv3] [TLSv1]** option specifies the SSL protocol versions that should not be used for client-side SSL negotiation.

If the **<clientssl client cert> <request|require|ignore>** option is set to request or require, all clients will be asked for a client certificate. If this option is omitted or set to ignore, the default is to disable requesting a client certificate. If this option is set to require, clients not presenting a valid and trusted client client certificate will not be permitted to establish a SSL connection. See also the **<clientssl ca file>** and **<clientssl client cert ca>** options.

If the **<clientssl authenticate> <once|always>** option is omitted or set to once, clients will be authenticated at most once for each SSL session. If this option is set to **always**, clients will be required to authenticate themselves (as directed by the **<clientssl client cert>** option) with each connection to the proxy.

The **<clientssl authenticate depth> <num>** option specifies the maximum number of certificates that will be traversed in a client certificate chain. If the certificate has not been verified in **<num>** steps, it will fail authentication. If this option is omitted, the default value is **9**.

The **<clientssl cache size> <num>** option specifies the maximum number of entries in the client-side SSL session cache. If this option is omitted, the default value is **20000**.

The **<clientssl cache timeout> <num>** option specifies the maximum lifetime of entries in the client-side SSL session cache in seconds. If this option is omitted, the default value is **300**.

The **<serverssl> <enable | disable>** option enables and disables the server-side SSL connection feature for the proxy being defined. If this option is omitted, the default is to have SSL disabled for all server-side connections.

The **<serverssl key> <serverside key file name>** option specifies a key file to be used as the private key corresponding to the server-side certificate file specified by the **<serverssl cert>** option. This option is not required, even when server-side SSL is enabled.

The **<serverssl cert> <serverside cert file name>** option specifies a certificate file to be used as the public key corresponding to the server-side key file specified by the **<serverssl key>** option. This certificate will be used as a client certificate when the proxy is asked to authenticate itself to servers. This option is not required, even when server-side SSL is enabled.

The **<serverssl chain> <serverside chain file name>** option specifies a chain file to be used to complete the certificate chain corresponding to the server-side certificate file specified by the **<serverssl cert>** option Certificates from this file will be used as necessary to build up the certificate chain. This option is not required, even when server-side SSL is enabled.

The **<serverssl ca file> <serverside CA file name>** option specifies a CA file to be used primarily to authenticate server certificates, but also to complete the certificate chain corresponding to the server-side certificate file specified by the **<serverssl cert>** option. See also the **<serverssl chain>** option. The CA file may contain more than one certificate. This option is not required, even when server-side SSL is enabled.

The **<serverssl ca path> <serverside CA path>** option specifies a path to a directory with certificate files to be used primarily to authenticate server certificates, but also to complete the certificate chain corresponding to the server-side certificate file specified by the **<serverssl cert>** options. See also the **<serverssl chain>** and **<serverssl ca file>** options. Unlike the **<serverssl ca file>** option, only the first certificate in certificate files with valid symbolic links generated by the **<make>** option are examined (the Makefile in the **/config/bigconfig/ssl.crt/ directory** should be used). This option is not required, even when server-side SSL is enabled.

The **<serverssl ciphers> <list>** option determines the set of ciphers available for server-side SSL negotiation. See the file **http://www.openssl.org/docs/apps/ciphers.html** for the format of **<list>**.

The <serverssl invalid> [SSLv2] [SSLv3] [TLSv1] option specifies the SSL protocol versions that should not be used for server-side SSL negotiation.

The **<serverssl server cert> <require|ignore>** option determines whether server certificates will be verified. If this option is set to **require**, all server certificates will be verified. If this option is set to **ignore**, server authentication only fails when a server presents an expired or malformed certificate. If this option is omitted, the default is to require verified server certificates. See also the **<serverssl ca file>** option.

The **<serverssl authenticate depth> <num>** option specifies the maximum number of certificates that will be traversed in a server certificate chain. If the certificate has not been verified in **<num>** steps, it fails authentication. If this option is omitted, the default value is **9**.

The **<akamaize> <enable | disable>** option enables Akamaization for the proxy being defined. If this option is omitted from the definition, then the default is to have Akamaization disabled.

The **<header insert> <quoted string>** option specifies a string to be prepended to the block of HTTP headers supplied with each client request. This string should take the standard HTTP header form of **"<field>:<value>"**.

The **<redirects rewrite> matching | all [enable]** option enables the rewriting of HTTP 301, 302, 303, 305, or 307 redirects' **Location** field to **"Location: https://..."**. When matching is specified, only a redirect with a URI in the **Location** field matching the URI requested by the client will be

rewritten. When all is specified, redirects are rewritten whether or not the URIs match. The port number specified in the redirect is also rewritten when it does not match the port of the proxy.

The **<lasthop pool> <none | lasthop pool name>** option specifies a lasthop pool to be used for the proxy. If this option is omitted, the default is to have no lasthop pool.

The **<arp> <enable | disable>** option enables and disables the **arp** for this proxy address. If this option is omitted, the default is to have **arp** enabled.

The **<vlans> <vlan name> [<vlan name>...] <enable|disable>** option enables and disables proxy access on existing VLANs. A proxy is accessible on all VLANs by default.

Creating a proxy server

The following example creates an SSL proxy:

```
bigpipe proxy 10.2.3.1:https target virtual 12.2.3.1:http \
key my.key cert my.crt
```

In this example, the BIG-IP system creates an SSL proxy, along with the key **my.key** and the certificate **my.crt**. As shown in the syntax listed above, many options are available when creating an SSL proxy, such as the server-side SSL proxy option and features related to client authentication.

Deleting a proxy server

You can delete an existing proxy server, using the following command-line syntax:

```
bigpipe proxy <ip>:<service> delete
```

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. For the **priority** load balancing mode, the command sets the priority level. Note that multiple node addresses can have the same priority level setting.

Options

The **<node_ip>** variable specifies an IP address of a specific node.

The **<weight>** variable specifies a whole number. The default weight for any node address is **1**.

The **show** option displays the ratio weights for the specified node addresses.

Displaying ratio settings

To display the current ratio settings for all node address that have ratio settings, use the following command:

```
b ratio [show]
```

The following output is displayed:

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

To display the ratio settings for specific node addresses, use the following command-line syntax:

```
b ratio <node addr> ... <node addr> [show]
```

Modifying ratio settings

The following command sets the ratio to **3** for the node address specified:

```
b ratio 192.168.103.20 3
```

reset

```
b reset
```

Clears the configuration values and counter values from memory.

Use this command with caution. All network traffic stops when you run this command.

Typically, this command is used on a standby BIG-IP system 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 systems:

```
b reset
b load <filename>
```

This sequence of commands ensures that only the values set in the **<filename>** specified are in use.

Note

This command does not delete any externally-stored classes. For information on how to delete externally-stored classes, see Chapter 5, iRules.

rule

```
b rule <rule_name> '{ if ( <expression> ) { <if statement> | <use pool statement> |
<discard statement> | <cache statement> | <redirect statement> | <log statement> |
<accumulate statement> | <hash statement> <if statement> } [ { else <statement> } ] [ { else
if <statement> } ] }'
b rule <rule_name> '{ discard }'
b rule <rule_name> '{ use pool <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> ] [ persist <expression> ]} }'
b rule <rule_name> '{ redirect <redirect URL> }'
b rule <rule name> '{ hash ( variable ) }'
b rule <rule_name> { if '( <statement> ) { use pool ( <statement> )' } }
b rule <rule_name> { if '( <statement> )' { use pool '( <statement> )' } else { '(
    <statement> )' } }
b rule <rule_name> { if '( <statement> )' { use pool '( <statement> )' } else { '(
    <discard_statement> )' } }
b rule <rule_name> { if '( <statement> )' { use pool '( <statement> )' } else { '(
    <redirect_statement> )' } }
b rule <rule_name> { if '( <statement> )' { use pool '( <statement> )' } else { '(
    <cache_statement> )' } }
b rule <rule_name> { if '( <statement> )' { use pool '( <statement> )' } else { '(
    <log_statement> )' } }
b rule <rule_name> { if '( <statement> )' { use pool '( <statement> )' } else { '(
    <accumulate_statement> )' } }
b rule <rule_name> delete
b rule <rule_name> show
```

Creates, deletes, or displays the rules on the BIG-IP system. Rules allow a virtual server to access any number of pools on the BIG-IP system. Based upon a simple or complex expression a pool can be selected through a rule. For more detailed information about using rules, see Chapter 5, *iRules*.

◆ Note

Before you define a rule, you must define the pool or pools that you want the rule to reference.

Rule statements

The **<rule_name>** variable specifies the name of a rule.

The **<pool_name>** variable specifies the name of a pool to be associated with the specified rule.

The **<if statement>** variable asks a true or false question and, depending on the answer, takes some action.

A **<discard statement>** variable discards the request. This statement must be conditionally associated with an **if** statement.

A **<use pool statement>** variable uses a selected pool for load balancing. This statement must be conditionally associated with an **if** statement.

A **<redirect statement>** variable sends traffic to a specific destination, rather than to a pool for load balancing.

A **<cache statement>** variable uses a selected pool for load balancing. This statement can be conditionally associated with an **if** statement. For attributes that you can use within a **cache** statement, see *Cache statement attributes*, on page A-58.

A **<log statement>** variable logs a message to the Syslog facility. The statement does this by performing variable expansion on the message as defined for **header_insert**.

An <**accumulate statement**> variable terminates rules processing until another another packet containing additional data is received from the originating client. This statement is useful with the **http_content** and **tcp_content** rule variables, when not enough data has been received to be successfully evaluated.

Cache statement attributes

Figure A.7 shows the attributes that can be used as arguments within **cache** statements.

Attribute	Description
origin_pool <pool_name>	This required attribute specifies a pool of servers with all the content to which requests are load balanced when the requested content is not cacheable or when all the cache servers are unavailable or when you use a BIG-IP system 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.

Table A.7 *Attributes for use within cache statements*

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.
persist <expression>	This optional attribute specifies an expression that will be evaluated and used to persist to the same node within the cache pool.

Table A.7 *Attributes for use within cache statements*

Functions

The iRules feature offers a set of functions that you can use within rule expressions. You can specify two kinds of functions within rules--functions that return a string, and functions that return a node name.

Functions that return a string

The following functions within expressions are used primarily to implement persistence within a pool. These functions therefore return a string that you specify. Table A.8 lists and describes these functions.

Function Name	Description
findstr()	Finds a string within another string and returns the string starting at the offset specified from the match. The **findstr()** function takes the following arguments: **findstr(<expr>, <string>, <offset>)** **findstr(<expr>, <string>, <offset>, <length>)** **findstr(<expr>, <string>, <offset>, <termchr>)**
substr()	Returns the string starting at the offset specified. The **substr()** function takes the following arguments: **substr(<expr>, <offset>)** **substr(<expr>, <offset>, <length>)** **substr(<expr>, <offset>, <termchr>)**
getfield()	Splits a string on a character and returns the string corresponding to the specific field. The **getfield()** function takes the following arguments: **getfield(<expr>, <split>, <fieldnum>)**
findclass()	Finds the member of a class that contains the result of the specified expression and returns that class member. The **findclass()** function takes the following arguments: **findclass(<expr>, <classname>)**

Table A.8 *Functions that return a string*

Function Name	Description
decode_uri()	Evaluates the expression and returns a string with any **%XX** escape sequences decoded as per HTTP escape sequences defined in RFC2396. The **decode_uri()** function takes the following arguments: **decode_uri(<expr>)**
domain()	Parses and returns up to the specified number of trailing parts of a domain name from the specified expression. The **domain()** function takes the following arguments: **domain(<expr>, <count>)**
imid()	Used to parse the **http_uri** variable for an i-mode identifier string that can be used for i-mode persistence. The **imid()** function takes no arguments and simply returns the string representing the i-mode identifier.

Table A.8 *Functions that return a string*

Functions that return a node name

The following functions within expressions are used to directly select a particular node (pool member) within a pool. Table A.9 lists and describes these functions.

Function Name	Description
node()	Returns a literal node address converted from either a string representation of an address and port or a literal number representing the node address as an integer. The **node()** function is designed primarily to be used with the persist expressions for selecting a node on which to persist. The **node()** function takes the following arguments: **node(<expr>)**
mappclass2node)	Represents a short-hand combination of the functions **findclass()**, **findstr()**, and **node().** The mapclass2node() function takes the following arguments: **mapclass2node(<expr>, <classname>, [<delim>])**
wlnode()	Returns a literal node address converted from either a string representation of an address and port, a literal number representing the node address as an interger, or a literal node address. The **wlnode()** function is designed primarily to be used with the persist expressions for selecting a node to which to persist. The **wlnode()** function takes the following arguments: **wlnode(<expr>)**

Table A.9 *Functions that return a node name*

Variable operands

Table A.10 lists the variable operands that can be used within rule statements. For more information, see Chapter 5, *iRules*.

IP Packet Header Variables	
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 String Variables	
http_method	The **http_method** variable s the action of the HTTP request. Common values are **GET** or **POST**.
http_uri	The **http_uri** variable s 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** variable is the HTTP protocol version string. Possible values are "**HTTP/1.0**" or "**HTTP/1.1**".
http_content([(<minlength>)]	The **http_content** variable evaluates the string following an HTTP content tag that you specify.
http_content_collected	The **http_content_collected** variable returns the amount of content that has currently been collected.
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.

Table A.10 *Variable operands for rule statements*

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) }**
TCP Request String Variables	
tcp_content	The **tcp_content** variable allows you to create a basic expression that load balances traffic based on arbitrary data within a TCP/IP connection.
tcp_bytes_collected	The **tcp_bytes_collected** variable returns the amount of content that has currently been collected.

Table A.10 *Variable operands for rule statements*

Binary Operators

The binary operators that can be used within rule statements are as follows:

- or
- and
- contains
- matches
- equals
- starts_with
- ends_with
- matches_regex
- one of
- redirect to

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 pool ( cgi_pool )
}
      else { use pool ( another_pool ) }
 }
```

Figure A.6 *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> | <log_stmt> | <accumulate_stmt> | <cache_stmt>
}'
```

Or, you can type the same rule using the following syntax:

```
b rule <name> if { '(<if_stmt>)' | '(<use_stmt>)' |
'(<discard_stmt>)' | '(<redirect_stmt>)' | '(<log_stmt>}'|
'(<accumulate_stmt>)' | '(<cache_stmt>)' }
```

For example:

```
b rule my_pool { if '(client_addr == 10.12.12.10)' { use pool
'(pool_A80)' } }
```

```
b rule your_pool '{ if (client_addr == 10.12.12.10) { use pool
(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
```

Deleting a rule

Delete a rule using the following format:

```
bigpipe rule cgi_rule delete
```

Displaying a rule

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 system 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 <ip_addr> vlan <vlan_name> [ netmask <ip_mask> ][ broadcast <broadcast_addr>]
    [unit <id>]
b self <ip_addr> vlan (vlangroup_name>
b self <ip_addr> floating enable | disable
b self <ip_addr> delete
b self <ip_addr> show
b self show
b self <ip_addr> snat automap enable | disable
```

Defines a self IP address on a BIG-IP system or 3-DNS Controller. A self IP address is an IP address mapping to a VLAN or VLAN group and their associated interfaces on a BIG-IP system or 3-DNS Controller. 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 system. Additional self IP addresses may be created for health checking, gateway failsafe, routing, or other purposes. These additional self IP addresses are created using the **self** command.

Options

The **<ip_addr>** variable specifies an IP address to assign to the BIG-IP system or 3-DNS Controller.

The **vlan <vlan_name | vlangroup_name>** option specifies the VLAN or VLAN group to which the self IP address is being assigned.

The **netmask <ip mask>** option specifies an IP mask used to set the network of the self IP address.

The **broadcast <broadcast_addr>** option specifies the broadcast address.

The **unit <id>**option specifies an optional unit ID, **1** or **2**. The default value is **1**.

The **floating** option enables or disables a floating self IP address.

The **snat automap** option enables or disables SNAT automapping on the specified self IP address. Once **snat automap** is enabled, the self IP address can be used as the translation address when SNAT automapping is enabled for a VLAN.

Creating self IP addresses

The following are examples of using the **bigpipe self** command to create self IP addresses:

```
b self 10.1.0.1 vlan external netmask 255.255.0.0
b self 10.2.0.1 vlan internal netmask 255.255.0.0
```

For a redundant configuration, the IP addresses that are shared by the two units are configured as floating IP addresses. For example:

```
b self 10.1.1.1 vlan external netmask 255.255.0.0 floating
enable
b self 10.2.1.1 vlan internal netmask 255.255.0.0 floating
enable
```

To create self IP addresses that are shared between the two units in an active-active configuration, assign a unit number to each self IP address, as in the following examples:

```
b self 10.1.1.1 vlan external netmask 255.255.0.0 unit 1\
floating enable
b self 10.1.1.2 vlan external netmask 255.255.0.0 unit 2\
floating enable
b self 10.2.1.1 vlan internal netmask 255.255.0.0 unit 1\
floating enable
b self 10.2.1.2 vlan internal netmask 255.255.0.0 unit 2\
floating enable
```

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. An idle connection is one in which no data has been received or sent for the number of seconds specified by the service timeout command.

The default timeout value for **tcp** services is **1005**, and **60** seconds for **udp** services. For idle connection reaping to be effective, you should set the timeout value to be greater than the configured timeout for the service daemons installed on your nodes.

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.

Options

The **<service>** variable specifies any valid port number, between **1** and **65535**, inclusive, or any valid service name in the **/etc/services** file.

The **<limit>** variable specifies the maximum number of simultaneous connections to be allowed to the service for all virtual servers. To turn off a connection limit for a service, specify a value of **0**.

The **<seconds>** variable specifies the number of seconds until a connection to the service times out.

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]
```

The **snat** command creates and deletes secure network address translation (SNATs), and displays information about them. A SNAT 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.

This command also allows you to set properties on a SNAT. A SNAT defines the relationship between an externally visible IP address, or translated address, and a group of internal IP addresses, or orginating address, of individual servers at your site.

Options

The **<orig addr>** variable specifies an originating IP address, that is, an address that is behind the BIG-IP system.

The **<trans addr>** variable specifies a translated IP address, that is, an address that is outside the BIG-IP system.

The **<ip addr>** variable can specify either an originating or a translated address.

The **<vlan name>** variable specifies the name of an existing VLAN on which access to the SNAT is enabled or disabled. By default, a SNAT is accessible on all VLANs.

The **<id>** variable specifies a unit ID, currently **1** or **2**. The default unit ID value is **1**.

The **<limit>** variable specifies a connection limit.

The **<seconds>** variable specifies the number of seconds for timeout.

The **auto** option enables SNAT automapping.

Defining a SNAT

SNATs map one or more originating addresses to a single translated address. Use the following syntax to define one or many orginating addresses to translated address maps:

```
b snat map <orig addr> [<orig addr>... ] to <trans addr>
```

For example, the following command maps a SNAT, which has two clients, to a single translated address:

```
b snat map 192.140.100.10 192.140.100.20 to 192.168.11.22
```

You can set the following properties on a SNAT:

- A connection limit (**limit** option)
- A **tcp** timeout value (**timeout tcp** option)
- A **udp** timeout value (**timeout udp** option)
- Connection mirroring (**mirror** option)
- ARP enable or disable
- A VLAN deny access list

Deleting SNAT

Use the following command-line syntax to permanently delete one or more SNATs from the BIG-IP system configuration:

```
b snat <ip addr>... <ip addr> delete
```

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
b stp show
```

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 system. 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.

Options

The **<stp_name>** variable specifies an arbitrary name for the spanning tree protocol (STP) domain.

The **<interface_name>** variable specifies an interface name, for example, **3.1.**

The **show** option displays the interfaces that make up the STP domain.

summary

```
b summary
```

Displays a summary of current usage statistics. The output display format for the **summary** command is shown in Figure A.7. You can find detailed descriptions of each of statistic displayed by the **summary** command in Chapter 17, *Administering the BIG-IP System*.

```
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 no handler deny                  = 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 A.7 *The **summary** output display*

For more information on the out put of the **bigpipe summary** command, see Chapter 17, *Administering the BIG-IP System*.

trunk

```
b trunk <controlling_if> define <if_list>
b trunk [<controlling_if>] show [verbose]
b trunk [<controlling_if>] stats reset
b trunk [<controlling_if>] delete
```

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, see Chapter 3, *Post-Setup Tasks*.

Options

The **<controlling link>** variable specifies the name of the interface chosen to be the controlling link for the trunk. Any attributes of the controlling link at layer 2 and above, such as membership in a VLAN, apply to the trunk.

The **<link>** variable specifies an interface name, for example **3.1**.

The **show** option displays information and statistics for the trunk, on a single line.

The **<verbose>** option, used with the **show** option, displays the information and statistics for the trunk in wordier form.

The **delete** option deletes the specified interface.

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 system. 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_port_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_log_level** command.

Table A.11 compares use of the **bigpipe verbose** command to use of the **bigpipe global verbose_log_level** command.

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 system 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 system 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_port_denial** **b verbose bigip_tcp_port_denial** Turns UDP and TCP port denial on for both virtual server and BIG-IP system addresses.	**b global verbose_log_level=15**

***Table A.11 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 system, the **bigpipe version** command displays the output shown in Figure A.8.

```
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 A.8 *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> svcdown_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 and deletes virtual servers, and displays information about them. This command also allows you to set properties on a virtual server, such as connection mirroring, connection limits, and timeouts.

A virtual server defines the relationships between an externally visible IP address that clients use to connect to your site, and the internal IP addresses of individual member servers that actually provide services for your site.

Options

The **<virtual addr>** variable specifies the IP address of the virtual server.

The **<virtual port>** variable specifies a port number or service name.

The **<bitmask>** variable specifies a number representing the bits that are the network part. of the virtual IP address.

The **<vlan name>** variable specifies the name of an existing VLAN for which you want to enable or disable access. By default, a virtual server is accessible on all VLANs.

The **<id>** variable specifies a unit id, currently **1** or **2**. The default value for the unit ID is **1**.

The **<ip addr>** variable specifies the IP address, of the form **10.20.30.40**.

The **use pool <pool name>** option specifies the name of an existing server pool.

The **<rule name>** variable specifies the name of an existing rule that the virtual server should reference.

The **translate port** option enables, disables, or shows port translation for a virtual server.

The **svcdown_reset** option enables, disables, or shows the ability of the BIG-IP system to automatically reset connections when a service becomes unavailable.

The **translate addr** option enables, disables, or shows address translation for a virtual server.

The **lasthop pool** option allows you to specify a pool to which to send connections back, instead of using the same router from which the connection was received.

The **mirror conn** option enables, disables, or shows the mirroring of connections in active/standby configurations.

The **conn rebind** option enables, disables, or shows dynamic connection rebinding.

The **stats reset** opton resets statistics for a virtual server.

The **accelerate** option enables, disables, or shows FastFlow acceleration, that is, increased speed of packet flow for TCP connections when the packets are not fragmented.

The **arp** option causes the BIG-IP system to respond to ARP requests for the virtual server address and send a gratuitous ARP request for router table updates.

The **any_ip** option allows you to configure a virtual server to load balance IP traffic other than TCP and UDP traffic, such as IPSEC traffic.

Defining a virtual server using pools and rules

To associate a pool of members with a virtual server, use a command such as the following:

```
b virtual 10.20.2.102:http use pool cgi_pool
```

To associate a rule with a virtual server, use a command such as the following:

```
b virtual 10.20.2.102:http use rule cgi_rule
```

Defining a virtual server with a wildcard port

Use the following command-line syntax to define an individual virtual server and the node or nodes to which the virtual server maps. Note that this syntax allows wildcard ports:

```
b virtual <virt addr> use pool <pool name> | rule <rule name>
```

You can also create multiple wildcard servers, one per VLAN. To create a wildcard server for a VLAN, use the following syntax:

```
bigpipe virtual <vlan_name> use pool <pool_name>
```

Deleting a virtual server

Use the following command-line syntax to permanently delete one or more virtual servers from the BIG-IP system configuration:

```
b virtual <virt addr>:<port>... <virt addr>:<port> delete
```

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> rename <new_vlan_name>
b vlan <vlan_name> mac_masq delete
b vlan <if_name> mac_masq <mac_addr> | show
b vlan <if_name> mac_masq 0:0:0:0:0
b vlan <vlan name> l2_agingtime <seconds>
b vlan <vlan name> fdb add <MAC address> interface <if_name>
b vlan <vlan name> fdb delete <MAC address> interface <if_name>
b vlan <vlan name> fdb static show
b vlan <vlan name> fdb dynamic show
b vlan <vlan name> fdb show
```

This command creates, displays and modifies settings for VLANs. VLANs are part of the base configuration.

When creating a VLAN, a tag value (VLAN ID) for the VLAN is automatically chosen unless it is specified on the command line. If a tag is specified on the command line and that tag number is **0**, the **vlan** command creates an empty VLAN.

A VLAN can have both tagged and untagged interfaces. An interface can be added to a single VLAN as an untagged interface. Also, an interface can be added to multiple VLANs as a tagged interface.

The **vlan** command defines VLANs, VLAN mappings, and VLAN properties. By default, each interface on a BIG-IP system or 3-DNS Controller 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.

Options

The **<vlan name>** variable specifies a VLAN name, 1-15 characters in length.

The **tag <tag number>** option specifies a valid VLAN tag number, in the range **0**-**4095**. Note that if **0** is specified as the tag number, the **vlan** command creates an empty VLAN.

The **interfaces add [tagged]** option specifies that the interfaces specified with the **<if_list>** argument are to be added to the specified VLAN, as either tagged or untagged interfaces.

The **interfaces delete** option deletes all interfaces for the specified VLAN.

The **<if_list>** variable specifies a list of interfaces to be added to a VLAN.

The **interfaces show all** option shows all interfaces for the specified VLAN.

The **port_lockdown** option enables or disables connection to the BIG-IP system through the specified VLAN.

The **proxy_forward** option enables or disables proxy forwarding, for purposes of L2 forwarding.

The **failsafe** option allows you to arm, disarm, or show the failsafe mechanism for redundant systems.

The **timeout <timeout>** option specifies a timeout value for the failsafe mechanism.

The **snat automap** option enables SNAT automapping for the specified VLAN.

The **rename <new_vlan_name>** option specifies the name to which you want to rename the specified VLAN.

The **<if_name>** variable specifies an interface name.

The **mac_masq <MAC address>** option specifies a MAC address, such as **0:a0:be:ef:1f:3a, that will be** shared by both BIG-IP units in a redundant system.

The **l2_agingtime <seconds>** option specifies the value in seconds of **L2 forwarding time**.

The **fdb** option adds the specified interfaces as entries in the L2 forwarding table.

The **port_lockdown** option enables or disables connection to the BIG-IP system through the specified VLAN.

vlangroup

```
b vlangroup [<vlan name list>] [show]
b vlangroup [<vlan name list>] list
b vlangroup <vlan name list> delete
b vlangroup <vlan name> tag <number>
b vlangroup [<vlan name list>] tag [show]
b vlangroup [<vlan name list>] interfaces [show]
b vlangroup <vlan name> vlans add <vlan if name list>
b vlangroup <vlan name list> vlans delete <vlan if name list>
b vlangroup <vlan name list> vlans delete all
b vlangroup [<vlan name list>] vlans [show]
b vlangroup <vlan name list> port_lockdown enable | disable
b vlangroup [<vlan name list>] port_lockdown [show]
b vlangroup <vlan name list> proxy_forward enable | disable
b vlangroup [<vlan name list>] proxy_forward [show] vlangroup <vlan name list> failsafe
    arm
b vlangroup <vlan name list> failsafe disarm
b vlangroup [<vlan name list>] failsafe [show]
b vlangroup <vlan name list> timeout <number>
b vlangroup [<vlan name list>] timeout [show] vlangroup <vlan name list> snat automap
    enable (deprecated)
b vlangroup <vlan name list> snat automap disable (deprecated)
b vlangroup <vlan name list> mac_masq <MAC addr>
b vlangroup [<vlan name list>] mac_masq [show]
b vlangroup <vlan name list> fdb add <MAC addr> interface <if name>
b vlangroup <vlan name list> fdb delete <MAC addr> interface <if name>
b vlangroup [<vlan name list>] fdb [show]
b vlangroup [<vlan name list>] fdb show static
b vlangroup [<vlan name list>] fdb show dynamic
b 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.

Options

For a description of the **bigpipe vlangroup** command options, see *Options*, on page A-80.

B

bigdb Configuration Keys

- Supported bigdb configuration keys

Supported bigdb configuration keys

The **bigdb** database contains configuration elements for the BIG-IP system. Configuration options that the **bigdb** databse supports include:

- Fail-over
- State mirroring
- Gateway failsafe pingers
- Configuration synchronization
- Interface related settings
- Health monitor settings

The **bigdb** 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 the current setting of a bigdb configuration key

To display the value of a **bigdb** 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 bigdb configuration key

To create (set) a **bigdb** configuration key, use the following syntax:

```
b db set <key>
```

To set a **bigdb** 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 bigdb configuration key

To unset a **bigdb** 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 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 command as **bigstart reinit sod**). This command forces the system to reread the **bigdb** database and use the new values. To run this command, type the following:

```
b failover init
```

Table B.1, on page B-3, lists the failover and cluster keys.

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 **1**. By default this is off and active-standby mode is used.
Local.Bigip.Failover.UnitId	This key is required. Each BIG-IP system 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 **0** (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 system synchronizes its time with the other. Set this key to **1** to turn off the time synchronization feature.
Common.Bigip.Failover.DbgFile	File into which **sod** logs the fail-over debug information.
Common.Bigip.Failover.PrintPeerState = 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 **0** (off).
Common.Bigip.Cluster.ActiveKeepAliveSec = 1	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 unit to be failed if no message is received within the timeout period. Used by network fail-over. Default is **3** seconds.

Table B.1 *The bigdb keys that store fail-over information*

StateMirror keys

StateMirror keys control state mirroring. If you change one of these values, you must perform the following reinitialization:

```
bigstart reinit sfd
```

Table B.2 lists the **StateMirror** keys.

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 has 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.
Common.Bigip.StateMirror.NoGC = 0	When the BIG-IP system receives a new connection, by default state mirroring causes mirrored data structures to be deleted. 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 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. You can read this data file with the **sfread** program.
Common.Bigip.StateMirror.Username	IP address of this BIG-IP system. 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 system listens for connections from the active unit. The default is **1028**.
Local.Bigip.StateMirror.Ipaddr	IP address of this BIG-IP system.
Local.Bigip.StateMirror.PeerIpaddr	IP address of the BIG-IP peer unit. Requires a value.

Table B.2 *The **bigdb** keys that store state mirroring information*

Using Gateway Pinger keys

The **GatewayPinger** keys control the gateway failsafe pinger. If you change one of these values, you must reinitialize the system as follows:

```
bigstart reinit bigd
```

Table B.3 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 system.
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 B.3 *The bigdb keys that store gateway pinger information*

bigd keys

The **bigd** keys control the health monitors. If you change one of these values, you must re-initialize the system as follows:

```
bigstart reinit bigd
```

Table B.4 lists the **bigd** keys.

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 to non-zero so that the 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 B.4 *The **bigdb** keys that store **bigd** information*

Other keys

Table B.5 lists other keys contained in the **bigdb** database, 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 (**wmi**, **real_server**, and **snmp_dca**) when set to "true" or "yes".
Common.Bigip.CORBA.IIOPPort ="683"	Default CORBA IIOP port used for LINK-IT bigapi.
Common.Bigip.CORBA.SSLPort ="684"	Default CORBA IIOP SSL port used for LINK-IT bigapi.
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 the LINK-IT **bigapi.**

Table B.5 *Other* ***bigdb*** *keys*

C

Configuration Files

- BIG-IP configuration files

BIG-IP configuration files

The following table lists the configuration files on the BIG-IP system.

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 system.
/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 system.
/etc/hosts.allow	Stores the IP addresses of workstations that are allowed to make administrative shell connections to the BIG-IP system.
/etc/hosts.deny	Stores the IP addresses of workstations that are allowed to make administrative shell connections to the BIG-IP system.
/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.

Glossary

A record

The **A** record is the ADDRESS resource record that a Link Controller returns to a local DNS server in response to a name resolution request. The **A** record contains a variety of information, including one or more IP addresses that resolve to the requested domain name.

Any IP Traffic

Any IP Traffic is a feature that allows the BIG-IP system to load balance protocols other than TCP and UDP.

alternate method

The alternate method specifies the load balancing mode that the Link Controller uses to pick a virtual server if the preferred method fails. See also *preferred method*.

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.

authentication

Authentication is the process of verifying a user's identity when the user is attempting to log on to a system.

authorization

Authorization is the process of identifying the level of access that a logged-on user has been granted to system resources.

big3d

The **big3d** utility is a monitoring utility that collects metrics information about paths between a BIG-IP system 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 system 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 system and hosts the Configuration utility.

bigpipe

The **bigpipe** utility provides command line access to the BIG-IP system.

bigtop

The **bigtop** utility is a statistical monitoring utility that ships on the BIG-IP system. 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*.

certificate verification

Certificate verification is the part of an SSL handshake that verifies that a client's SSL credentials have been signed by a trusted certificate authority.

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.

clone pool

This feature causes a pool to replicate all traffic coming into it and send that traffic to another pool cloned from the first pool.

completion rate

The completion rate is the percentage of packets that a server successfully returns during a given session.

Completion Rate mode

The Completion Rate mode is a dynamic load balancing mode that distributes connections based on which network path drops the fewest packets, or allows the fewest number of packets to time out.

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.

content switching

Content switching is the ability to load balance traffic based on data contained within a packet.

cookie persistence

Cookie persistence is a mode of persistence where the BIG-IP system 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 system 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.

domain name

A domain name is a unique name that is associated with one or more IP addresses. Domain names are used in URLs to identify particular Web pages. For example, in the URL **http://www.f5.com/index.html**, the domain name is **f5.com**.

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 system. 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 system. See also *health check*.

external authentication

External authentication refers to the process of using a remote LDAP or RADIUS server to store data for the purpose of authenticating users attempting to log on to the BIG-IP system.

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 system. 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 load balancing method that passes a new connection based on the fastest response of all currently active nodes.

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.

Global Availability mode

Global Availability mode is a static load balancing mode that bases connection distribution on a particular server order, always sending a connection to the first available server in the list. This mode differs from Round Robin mode in that it searches for an available server always starting with the first server in the list, while Round Robin mode searches for an available server starting with the next server in the list (with respect to the server selected for the previous connection request).

health check

A health check is a BIG-IP system feature that determines whether a node is **up** or **down**. Health checks are implemented through health monitors. See also *health monito*r, *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 system is called an interface. See also *link*.

internal VLAN

The internal VLAN is a default VLAN on the BIG-IP system. 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.

Key Management System

The Key Management System (KMS) is a set of screens within the Configuration utility that allows you to centrally manage SSL proxy keys and certificates. You can generate certificate requests, install keys, and export and import keys and key archives. You can also associate keys with SSL proxies.

Kilobytes/Second mode

The Kilobytes/Second mode is a dynamic load balancing mode that distributes connections based on which available server currently processes the fewest kilobytes per second.

last hop

A last hop is the final hop a connection took to get to the BIG-IP system. You can allow the BIG-IP system 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 system 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.

local DNS

A local DNS is a server that makes name resolution requests on behalf of a client. With respect to the Link Controller, local DNS servers are the source of name resolution requests. Local DNS is also referred to as LDNS.

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.

metrics information

Metrics information is the data that is typically collected about the paths between Link Controllers and local DNS servers. Metrics information is also collected about the performance and availability of virtual servers. Metrics information is used for load balancing, and it can include statistics such as round trip time, packet rate, and packet loss.

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 system 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 system 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 system 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.

name resolution

Name resolution is the process by which a name server matches a domain name request to an IP address, and sends the information to the client requesting the resolution.

name server

A name server is a server that maintains a DNS database, and resolves domain name requests to IP addresses using that database.

NAT (Network Address Translation)

A NAT is an alias IP address that identifies a specific node managed by the BIG-IP system 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 system.

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 system uses to verify the status of multiple nodes. When the BIG-IP system uses a node alias to check node status, it pings the node alias. If the BIG-IP system receives a response to the ping, it marks all nodes associated with the node alias as **up**. If the BIG-IP system 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 system 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.

path

A path is a logical network route between a Link Controller and a local DNS server.

path probing

Path probing is the collection of metrics data, such as round trip time and packet rate, for a given path between a requesting LDNS server and a Link Controller.

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 system 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 system 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.

pool ratio

A pool ratio is a ratio weight applied to pools in a wide IP. If the Pool LB mode is set to Ratio, the Link Controller uses each pool for load balancing in proportion to the weight defined for the pool.

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.

preferred method

The preferred method specifies the first load balancing mode that the Link Controller uses to load balance a resolution request. See also *alternate method*.

QOS equation

The QOS equation is the equation on which the Quality of Service load balancing mode is based. The equation calculates a score for a given path between a link and a local DNS server. The Quality of Service mode distributes connections based on the best path score for an available link. You can apply weights to the factors in the equation, such as round trip time and completion rate.

Quality of Service load balancing mode

The Quality of Service load balancing mode is a dynamic inbound load balancing mode that bases connection distribution on a configurable combination of the packet rate, completion rate, round trip time, hops, virtual server capacity, kilobytes per second, and topology information.

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 system 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 system 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.

resource record

A resource record is a record in a DNS database that stores data associated with domain names. A resource record typically includes a domain name, a TTL, a record type, and data specific to that record type. See also *A record.*

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.

round trip time (RTT)

Round trip time is the calculation of the time (in microseconds) that a local DNS server takes to respond to a ping issued by the **big3d** agent running on a link. The Link Controller takes RTT values into account when it uses dynamic load balancing modes.

Round Trip Time mode

Round Trip Time mode is a dynamic load balancing mode that bases connection distribution on which virtual server has the fastest measured round trip time between the link and the local DNS server.

self IP address

Self IP addresses are the IP addresses owned by the BIG-IP system that you use to access the internal and external VLANs.

send string

A send string is the request that the BIG-IP system 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 system. 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 system 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 system.

SSL proxy

An SSL proxy is a gateway for decrypting HTTP requests to an HTTP server and encrypting the reply.

SSL-to-Server

SSL-to-Server is an SSL proxy feature that provides secure communication between the BIG-IP system and a target content server.

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 system 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 system. 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 system.

trunk

A trunk is a combination of two or more interfaces and cables configured as one link. See also *link aggregation.*

unavailable

The **unavailable** status is used for links and virtual servers. When a link or virtual server is **unavailable**, the Link Controller does not use it for load balancing.

unknown

The **unknown** status is used for links and virtual servers. When a link or virtual server is new to the Link Controller and does not yet have metrics information, the Link Controller marks its status as **unknown**. The Link Controller can use unknown servers for load balancing, but if the load balancing mode is dynamic, the Link Controller uses default metrics information for the unknown server until it receives live metrics data.

up

The **up** status is used for links and virtual servers. When a link or virtual server is **up**, the link or virtual server is available to respond to process connections.

Universal Inspection Engine

The Universal Inspection Engine (UIE) is a feature that offers universal persistence and universal content switching, to enhance your load balancing capabilities. The UIE contains a set of rule variables and functions for building expressions that you can specify in pool definitions and rules.

universal persistence

Universal persistence gives you the ability to persist on any string found within a packet. Also, you can directly select the pool member to which you want to persist.

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 system.

virtual port

A virtual port is the port number or service name associated with one or more virtual servers managed by the BIG-IP system. 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 system 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 system for hardware failure.

wide IP

A wide IP is a collection of one or more domain names that maps to one or more groups of virtual servers managed by Link Controllers. The Link Controller load balances name resolution requests across the virtual servers that are defined in the wide IP that is associated with the requested domain name.

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

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W