Configuration Guide for Local Traffic Management

version 9.2.2

MAN-0182-01

Product Version

This manual applies to version 9.2.2 of BIG-IP® Local Traffic Manager™, BIG-IP® Load Balancer Limited™, and BIG-IP® SSL Accelerator™.

Publication Date

This guide was published on January 12, 2006.

Legal Notices

Canadian Regulatory Compliance

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

Standards Compliance

This product conforms to the IEC, European Union, ANSI/UL and Canadian CSA standards applicable to Information Technology products at the time of manufacture.

Acknowledgments

This product includes software developed by Bill Paul.

This product includes software developed by Jonathan Stone.

This product includes software developed by Manuel Bouyer.

This product includes software developed by Paul Richards.

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

This product includes software developed by the Politecnico di Torino, and its contributors.

This product includes software developed by the Swedish Institute of Computer Science and its contributors.

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 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 the University of Vermont and State Agricultural College and Garrett A. Wollman.

This product includes software developed by Bal·zs Scheidler <bazsi@balabit.hu>, which is protected under the GNU Public License.

This product includes software developed by Niels M^ller <nisse@lysator.liu.se>, which is protected under the GNU Public License.

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).

Table of Contents

1
Introducing Local Traffic Management

2
Configuring Virtual Servers

3
Configuring Nodes

4
Configuring Load Balancing Pools

5
Understanding Profiles

6
Managing HTTP and FTP Traffic

7
Managing SSL Traffic

8
Authenticating Application Traffic

11
Configuring SNATs and NATs

12
Configuring Rate Shaping

13
Writing iRules

1

Introducing Local Traffic Management

- Introducing the BIG-IP system
- Understanding BIG-IP local traffic management
- Overview of local traffic management configuration
- About this guide
- Finding help and technical support resources

Introducing the BIG-IP system

The BIG-IP® system is a port-based, multilayer switch that supports virtual local area network (VLAN) technology. Because hosts within a VLAN can communicate at the data-link layer (Layer 2), a BIG-IP system reduces the need for routers and IP routing on the network. This in turn reduces equipment costs and boosts overall network performance. At the same time, the BIG-IP system's multilayer capabilities enable the system to process traffic at other OSI layers. The BIG-IP system can perform IP routing at Layer 3, as well as manage TCP, UDP, and other application traffic at Layers 4 through 7. The following modules provide comprehensive traffic management and security for many traffic types. The modules are fully integrated to provide efficient solutions to meet any network, traffic management, and security needs.

- **BIG-IP® Local Traffic Manager**
  The BIG-IP system includes local traffic management features that help make the most of network resources. Using the powerful Configuration utility, you can customize the way that the BIG-IP system processes specific types of protocol and application traffic. By using features such as virtual servers, pools, and profiles, you ensure that traffic passing through the BIG-IP system is processed quickly and efficiently, while meeting all of your security needs. For more information, see the ***Configuration Guide for Local Traffic Management***.

- **BIG-IP® Global Traffic Manager**
  The Global Traffic Manager provide intelligent traffic management to your globally available network resources. Through the Global Traffic Manager, you can select from an array of load balancing modes, ensuring that your clients access the most responsive and robust resources at any given time. In addition, the Global Traffic Manager provides extensive monitoring capabilities so the health of any given resource is always available. For more information, see the ***Configuration Guide for Global Traffic Management***.

- **BIG-IP® Link Controller**
  The Link Controller seamlessly monitors availability and performance of multiple WAN connections to intelligently manage bi-directional traffic flows to a site - providing fault tolerant, optimized Internet access regardless of connection type or provider. The Link Controller ensures that traffic is always sent over the best available link to maximize user performance and minimize bandwidth cost to a data center. For more information, see the ***Configuration Guide for the BIG-IP Link Controller***.

- **BIG-IP®Application Security Module**
  The Application Security Module provides web application protection from application-layer attacks. The Application Security Module protects Web applications from both generalized and targeted application layer attacks including buffer overflow, SQL injection, cross-site scripting, and parameter tampering. For more information, see the ***Configuration Guide for the BIG-IP Application Security Module***.

Understanding BIG-IP local traffic management

The BIG-IP® local traffic management (LTM) system is specifically designed to manage your local network traffic. ***Local traffic management*** refers to the process of managing network traffic that comes into or goes out of a local area network (LAN), including an intranet.

This configuration guide applies to the set of local traffic management products that are part of the BIG-IP® family of products.

A commonly-used feature of the LTM system is its ability to intercept and redirect incoming network traffic, for the purpose of intelligently tuning the load on network servers. However, tuning server load is not the only type of local traffic management. The LTM system includes a variety of features that perform functions such as inspecting and transforming header and content data, managing SSL certificate-based authentication, and compressing HTTP responses. In so doing, the LTM system not only directs traffic to the appropriate server resource, but also enhances network security and frees up server resources by performing tasks that web servers typically perform.

Summary of local traffic-management capabilities

When configured properly, the LTM system can perform a wide variety of traffic-management functions, such as:

- Balancing traffic to tune and distribute server load on the network for scalability.
- Off-loading standard server tasks, such as HTTP data compression, SSL authentication, and SSL encryption to improve server performance.
- Monitoring the health and performance of servers on the network for availability.
- Establishing and managing session and connection persistence.
- Handling application-traffic authentication and authorization functions based on user name/password and SSL certificate credentials.
- Managing packet throughput to optimize performance for specific types of connections.
- Improving performance by aggregating multiple client requests into a server-side connection pool. This aggregation of client requests is part of the LTM system's OneConnect™ feature.
- Applying configuration settings to customize the flow of application-specific traffic (such as HTTP and SSL traffic).
- Customizing the management of specific connections according to user-written scripts based on the industry-standard Tool Command Language (Tcl).

While some of the functions on this list offer the basic ability to balance the load on your network servers, other functions on the list offer specialized abilities that are worth noting. These abilities include managing specific types of application traffic, optimizing server performance, and enhancing the security of your network. The following sections describe these specialized capabilities.

Managing specific types of application traffic

Applying configuration settings to customize the flow of application-specific traffic is a key feature of local traffic management. The LTM system can control many different kinds of traffic, each in a different way. You do this by establishing a policy for managing each type of network traffic. Examples of traffic types that the system can manage are: TCP, UDP, HTTP, FTP, SSL, Session Initiation Protocol (SIP), i-mode®, and Microsoft® Remote Desktop Protocol (MSRDP).

In addition to creating separate policies to systematically manage these different traffic types, you can also do the following:

- Write iRules™ to assign certain behaviors to individual application-specific connections. iRules can search the content of a particular type of traffic, such as an HTTP request or response, and direct the traffic accordingly.
- Insert header data into application-specific requests, such as HTTP requests, and then direct the request based on that header data.
- Implement session persistence. Using the LTM system's powerful configuration tools, you can configure session persistence, based on data such as HTTP cookies, source IP addresses, destination IP addresses, and SSL session IDs.
- Monitor the health or performance of servers in a pool. For example, the LTM system can monitor Lightweight Directory Access Protocol (LDAP) servers on a network, and if the system determines that a target LDAP server is non-functional, the LTM system can redirect the request to a different LDAP server.
- Use the dynamic ratio load-balancing algorithm to assess the current load on a particular type of server, such as a Windows Management Infrastructure (WMI) server, and then redirect a request based on that assessment. The ability to monitor servers corresponding to specific types of applications is a key tool for maintaining optimal performance of your network.

Optimizing performance

The LTM system includes several features designed to optimize server performance. Such features either offload labor-intensive traffic management tasks, such as SSL certificate verification, or enable the pooling, reuse, and overall persistence of server-side connections.

Offloading server tasks

The tasks that the LTM system can offload from a network server are:

- SSL certificate-based authentication, including the checking of certificate revocation status through OCSP
- SSL encryption and decryption
- SSL certificate-based authorization using remote LDAP servers
- HTTP data compression
- The rewriting of MSRDP connections

Optimizing TCP and HTTP connections

The LTM system manages TCP and HTTP connections in certain ways to optimize server performance. Primary network optimization features are: OneConnect™, HTTP pipelining, and rate shaping.

OneConnect

The OneConnect™ feature contains the following components:

- **Content Switching**
  When an HTTP client sends multiple requests within a single connection, the LTM system is able to process each of those requests individually, sending those requests to different destination servers if necessary. This feature is enabled automatically and does not require configuration.
- **Connection Pooling**
  With this feature, the LTM system combines server-side connections that are not in use, so that other clients can use them. This can significantly reduce the number of servers required to process client requests. By default, this feature is disabled, but can be easily enabled using a OneConnect profile.
- **OneConnect transformation**
  Sometimes, for HTTP/1.0 requests, you might want to add **Keep-Alive** support to HTTP **Connection** headers, to ensure that server-side connections remain open. This manipulation of HTTP **Connection** headers is a feature known as OneConnect transformation. This feature works best when used in conjunction with connection pooling.

For more information on OneConnect™, see Chapter 5, *Understanding Profiles*, and Chapter 6, *Managing HTTP and FTP Traffic*.

HTTP pipelining

In addition to the OneConnect™ feature, the LTM system has the ability to process pipelined requests. This means that the LTM system can process a client request even if the previous request has not yet received a response. Pipelining is an optimization feature available for HTTP/1.1 requests only.

For more information on HTTP pipelining, see Chapter 6, *Managing HTTP and FTP Traffic*.

Rate shaping

Rate shaping is a feature that allows you to categorize certain types of connections into rate classes, for the purpose of customizing the throughput of those connections. This is useful, for example, when you want to optimize web-server performance for preferred Internet customers.

TCP optimizations

The LTM system includes significant TCP optimizations, such as in-order delivery and content spooling.

Enhancing network security

Security is an important consideration in managing local network traffic. Accordingly, the LTM system contains a number of features designed to assist in preventing security breaches. These features pertain not only to authenticating and authorizing users and applications, but also to detecting intrusions and mitigating DOS attacks.

In general, when the LTM system detects a security problem, it can take actions such as:

- Reject a client request based on SSL certificate verification
- Reject and discard unauthorized packets
- Alert system administrators to an attack or infiltration attempt
- Direct suspicious traffic to specific target servers
- Log authentication failures
- Prevent SYN flooding

An important consideration for any networked environment is the authentication and authorization mechanism that you use to authenticate users and their client requests and to control user and application access to server resources. To this end, the LTM system supports Pluggable Authentication Module (PAM) technology, and provides a complete set of PAM authentication modules that you can choose from to handle your authentication or authorization needs.

The authentication modules that the LTM system provides are as follows:

- **An LDAP module**
  Uses a remote LDAP server to perform user name/password user authentication.
- **A RADIUS module**
  Uses a Remote Authentication Dial In User Service (RADIUS) server to perform user name/password user authentication.
- **A TACACS+ module**
  Uses a remote Terminal Access Controller Access Control System (TACACS+) server to perform user name/password user authentication.
- **An SSL Client Certificate LDAP module**
  Uses a remote LDAP server to perform SSL certificate-based authorization of client SSL traffic.
- **An OCSP module**
  Uses a remote Online Certificate Status Protocol (OCSP) server to provide up-to-date SSL certificate revocation status for the purpose of authenticating client and server SSL traffic.

Overview of local traffic management configuration

Once you have set up your base network and you have administrative access to the LTM system, and at least a default VLAN assignment for each interface, the next step is to configure a network for managing traffic targeted to your internal servers.

At the heart of the LTM system are virtual servers and load balancing pools. Virtual servers receive incoming traffic, perform basic source IP and destination IP address translation, and direct traffic to servers, which are grouped together in load balancing pools.

To configure a basic local traffic management system, you use the Configuration utility. With this utility, you can create a complete set of configuration objects that work together to perform local traffic management. Each object has a set of configuration settings that you can use as is or change to suit your needs. These objects are:

- **Virtual servers**
  Virtual servers receive requests and distribute them to pool members.
- **Nodes**
  Nodes represent server IP addresses on your network that you can enable and disable, and for which you can obtain status.
- **Load balancing pools**
  Load balancing pools contain servers to which requests can be sent for processing.
- **Application-type profiles**
  Application-type profiles contain settings that define the behavior of various traffic types, such as TCP, HTTP, and SSL.
- **SSL Certificates**
  The SSL Certificates object allows you to generate SSL certificate requests and install SSL certificates on the LTM system, for the purpose of terminating and initiating SSL connections.
- Remote authentication
  The remote authentication feature allows you to use a remote server to authenticate application traffic. Examples of remote authentication servers are LDAP, RADIUS, and OCSP servers.
- **Session Persistence profiles**
  Session persistence profiles allow you to implement session persistence based on a variety of criteria such as HTTP cookies, source IP addresses, and destination IP addresses.
- **Monitors**
  Monitors track the current health or performance of pool members.
- **SNATs**
  Secure Network Address Translations (SNATs) translate the source IP address in a client request, allowing multiple hosts to share the same address.

- **Rate Shaping**
  Rate shaping controls bandwidth consumption.
- **iRules**
  iRules can define criteria for pool-member selection, as well as perform content transformations, logging, custom protocol support, and so on.

When you create configuration objects, you can choose to perform either basic or advanced configuration:

- **Basic**
  You choose a ***basic configuration*** when you want to primarily use the default values for your object settings. When you choose a basic configuration, the Configuration utility displays only those few settings that you would most likely need to modify. The other settings remain hidden and retain their default values. Choosing a basic configuration is an easy way to create configuration objects.
- **Advanced**
  You choose an ***advanced configuration*** when you want to modify many of the values for your object settings. When you choose an advanced configuration, the Configuration utility displays all of the object's settings and allows you to modify any of them.

The three most important objects in the LTM system that you must configure for local traffic management are:

- Virtual servers
- Load balancing pools
- Profiles

Configuring virtual servers

When you create a virtual server, you specify the type of virtual server you want, that is, a host virtual server or a network virtual server. Then you can attach various properties and resources to it, such as application-specific profiles, session persistence, and user-written scripts called iRules that define pool-selection criteria. All of these properties and resources, when associated with a virtual server, determine how the LTM system manages local traffic.

When you create and configure a virtual server, you use the part of the Configuration utility screen shown in Figure 1.1, on page 1-9.

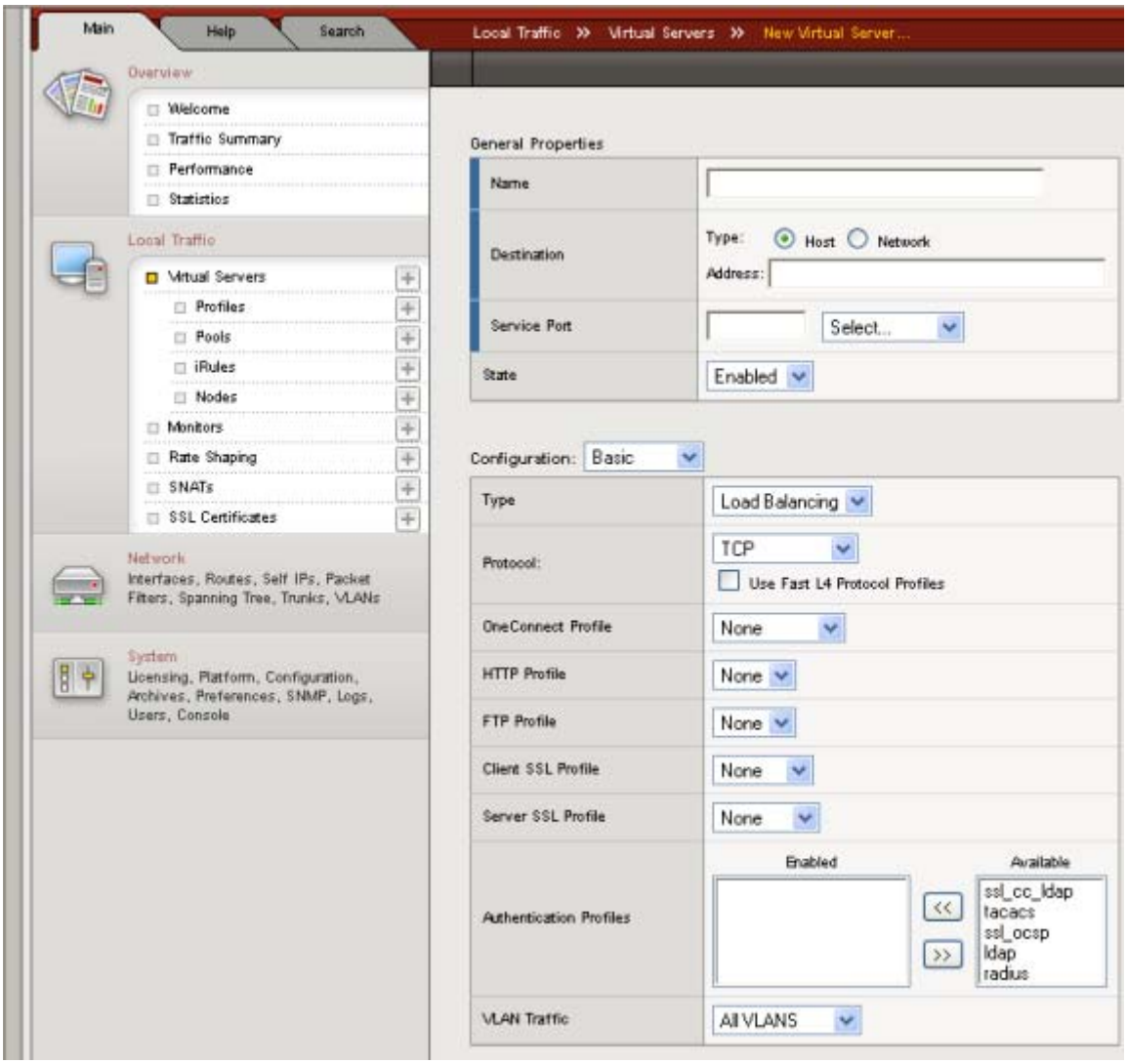

Figure 1.1 *The Configuration utility screen for creating a virtual server*

For more information on virtual servers, see Chapter 2, *Configuring Virtual Servers*.

Configuring load balancing pools

A ***load balancing pool*** is a collection of internal servers that you group together to service client requests. A server in a pool is referred to as a ***pool member.*** Using the default load balancing algorithm, known as Round Robin, the LTM system sends a client request to a member of that pool.

To implement a load balancing pool, you first create the pool, and then you associate the pool name with an existing virtual server. A virtual server sends client requests to the pool or pools that are associated with it. The virtual server screen shown in Figure 1.1, on page 1-9 includes a setting, **Default Pool**, for specifying a pool name.

Pools have settings associated with them, such as IP addresses for pool members, load balancing modes, and health and performance monitors. When you create a pool, you can use the default values for some of these settings, or change them to better suit your needs.

When you create and configure a load balancing pool, you use the Pool screen of the Configuration utility. Figure 1.2 shows part of this screen.

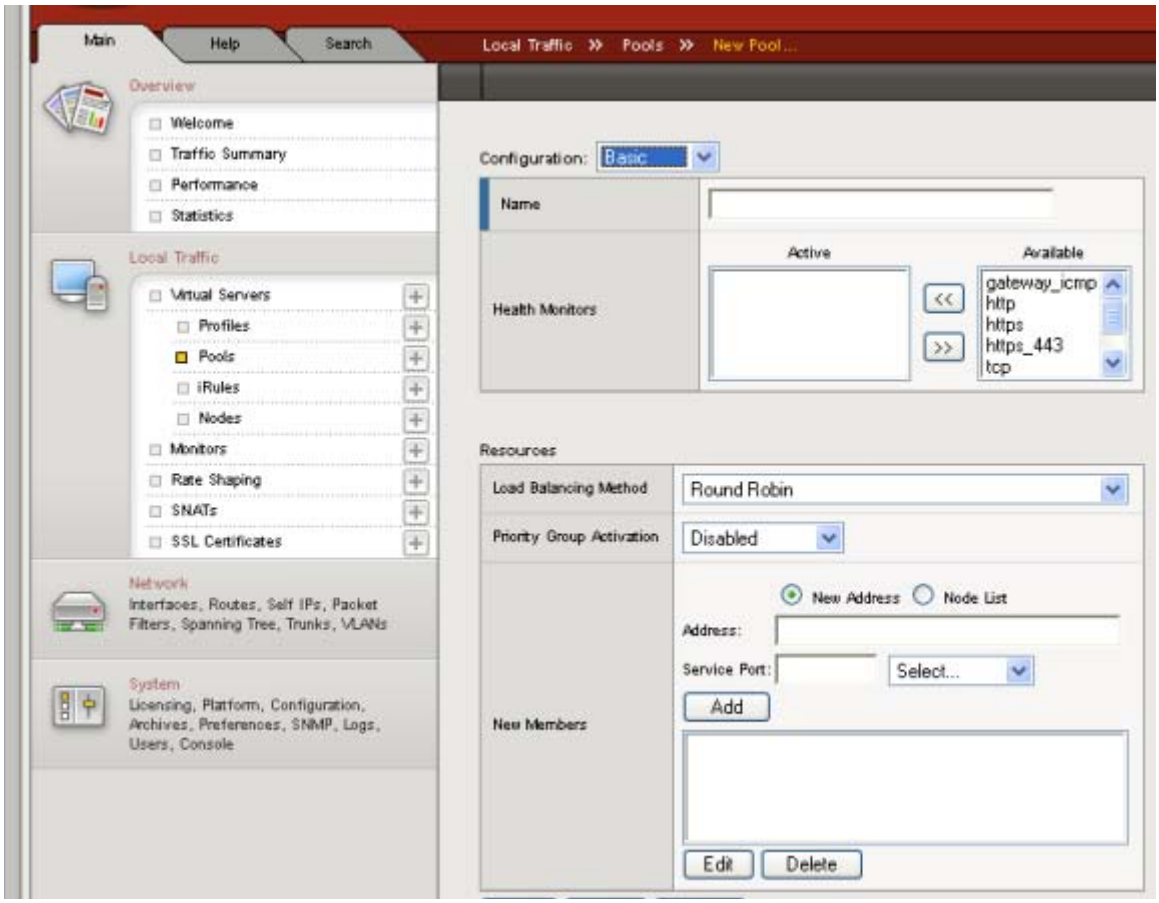

Figure 1.2 *The Configuration utility screen for creating a load balancing pool*

For more information on load balancing pools, see Chapter 4, *Configuring Load Balancing Pools*.

Configuring profiles

A ***profile*** is a group of configuration settings that apply to a specific type of network traffic, such as HTTP connections. If you want the virtual server to manage a type of traffic, you can associate the applicable profile with the virtual server, and the virtual server applies that profile's settings to all traffic of that type.

For example, you might want the LTM system to compress HTTP response data. In this case, you can configure an HTTP profile to enable compression, and associate the profile with a virtual server. Then, when the virtual server processes an HTTP request, the LTM system compresses the response.

There are several types of profiles that you can create for your own needs. They are: FastL4, TCP, UDP, One Connect, Stream, HTTP, FTP, Client SSL, Server SSL, Persistence, and Authentication. When you create a profile, you can use the default values for the settings, or change them to better suit your needs.

For example, when you create and configure an HTTP profile, you use the part of the Configuration utility screen shown in Figure 1.3, on page 1-12.

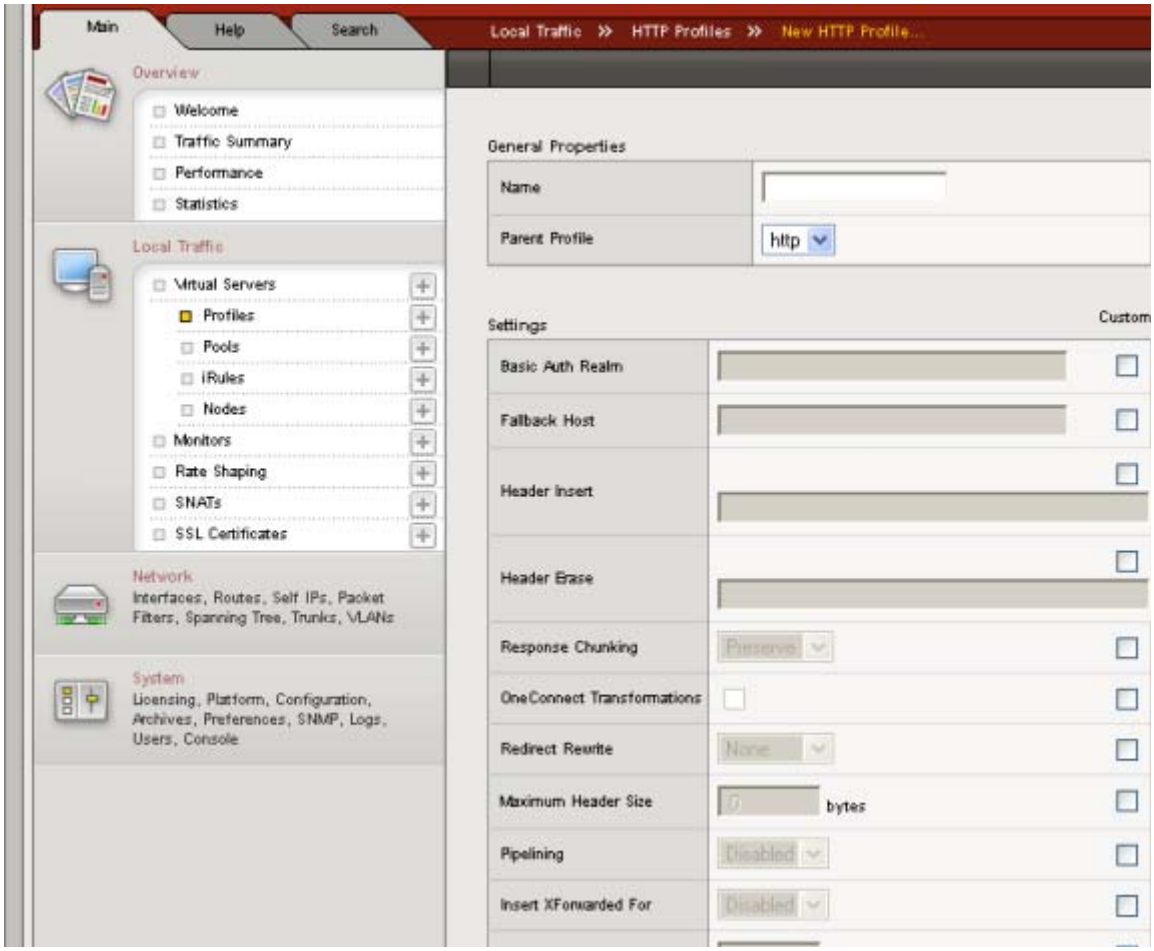

Figure 1.3 *The Configuration screen for creating an HTTP profile*

For more information on configuring profiles, see Chapter 5, *Understanding Profiles*, and one of the following chapters:

- Chapter 6, *Managing HTTP and FTP Traffic*
- Chapter 7, *Managing SSL Traffic*
- Chapter 8, *Authenticating Application Traffic*
- Chapter 9, *Enabling Session Persistence*

Using the Configuration utility

All users need to use the web-based Configuration utility in order to license the system for the first time.

In addition to setting up the management network and initial traffic management software configuration, you use the Configuration utility to configure and monitor the LTM system. You can use the Configuration

utility to perform additional configuration steps necessary for your configuration. In the Configuration utility, you can also monitor current system performance. Most procedures in this guide use the Configuration utility.

The Configuration utility supports Netscape® Navigator™, version 7.1, or other browsers built on the same engine, such as Mozilla™, Firefox™, and Camino™; and Microsoft® Internet Explorer™ version 6.x and later.

For information on setting user preferences for the Configuration utility, see the ***Network and System Management Guide***.

About this guide

This guide describes how to configure the BIG-IP local traffic management system to manage traffic coming into, or leaving, the local traffic network. Before you can configure the features described in this guide, you must install the BIG-IP system, license the system, and use the Setup utility to set up the network configuration.

Additional information

In addition to this guide, there are other sources of the documentation you can use in order to work with the BIG-IP system. The information is organized into the guides and documents described below. The following printed documentation is included with the BIG-IP system.

- **Configuration Worksheet**
  This worksheet provides you with a place to plan the basic configuration for the BIG-IP system.
- **BIG-IP Quick Start Instructions**
  This pamphlet provides you with the basic configuration steps required to get the BIG-IP system up and running in the network.

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 system. It also contains important environmental warnings.
- **Installation, Licensing, and Upgrades for BIG-IP Systems**
  This guide provides detailed information about installing upgrades to the BIG-IP system. It also provides information about licensing the BIG-IP system software and connecting the system to a management workstation or network.
- **Network and System Management Guide**
  This guide contains any information you need to configure and maintain the network and system-related components of the BIG-IP system. With this guide, you can perform tasks such as configuring VLANs, assigning self IP addresses, creating administrative user accounts, and managing a redundant system.

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, you can set the **Idle Timeout** value to **5**.

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, for installation instructions, refer to Chapter 1, *Installing the Software*, in the ***Installation, Licensing, and Upgrades for BIG-IP Systems*** guide.

Finding help and technical support resources

You can find additional technical documentation and product information in the following locations:

- **Online help for local traffic management**
  The Configuration utility has online help for each screen. The online help contains descriptions of each control and setting on the screen. Click the Help tab in the left navigation pane to view the online help for a screen.
- **Welcome screen in the Configuration utility**
  The Welcome screen in the Configuration utility contains links to many useful web sites and resources, including:
  - The F5 Networks Technical Support web site
  - The F5 Solution Center
  - The F5 DevCentral web site
  - Plug-ins, SNMP MIBs, and SSH clients
- **F5 Networks Technical Support web site**
  The F5 Networks Technical Support web site, **http://tech.f5.com**, provides the latest documentation for the product, including:
  - Release notes for the <product names>, current and past
  - Updates for guides (in PDF form)
  - Technical notes
  - Answers to frequently asked questions
  - The Ask F5 natural language question and answer engine.

  To access this site, you need to register at **http://tech.f5.com**.

2

Configuring Virtual Servers

- Introducing virtual servers
- Understanding virtual server types
- Creating and modifying virtual servers
- Configuring virtual server and virtual address settings
- Managing virtual servers and virtual addresses

Introducing virtual servers

Virtual servers are the most important component of any BIG-IP® local traffic management (LTM) configuration. A ***virtual server*** receives a client request, and instead of sending the request directly to the destination IP address specified in the packet header, sends it to any of several content servers that make up a load balancing pool. Virtual servers increase the availability of resources for processing client requests.

Not only do virtual servers distribute traffic across multiple servers, they also treat varying types of traffic differently, depending on your traffic-management needs. For example, a virtual server can enable compression on HTTP request data as it passes through the LTM system, or decrypt and re-encrypt SSL connections and verify SSL certificates. For each type of traffic, such as TCP, UDP, HTTP, SSL, and FTP, a virtual server can apply an entire group of settings, to affect the way that the LTM system manages that traffic type.

A virtual server can also enable session persistence for many different traffic types. Through a virtual server, you can set up session persistence for HTTP, SSL, SIP, and MSRDP connections, to name a few.

Finally, a virtual server can apply an iRule, which is a user-written script designed to inspect and direct individual connections in specific ways. For example, you can create an iRule that searches the content of a TCP connection for a specific string and, if found, directs the virtual server to send the connection to a specific pool or pool member.

To summarize, a virtual server can do the following:

- Distribute client requests across multiple servers to balance server load
- Apply various behavioral settings to multiple traffic types
- Enable persistence for multiple traffic types
- Direct traffic according to user-written iRules™

You can use virtual servers in any of several distinct ways:

- **Directing traffic to a load balancing pool**
  A ***Standard*** virtual server (also known as a ***load balancing*** virtual server) directs client traffic to a load balancing pool and is the most basic type of virtual server. When you first create the virtual server, you assign an existing default pool to it. From then on, the virtual server automatically directs traffic to that default pool.
- **Sharing an IP address with a VLAN node**
  You can set up a ***Forwarding (Layer 2)*** virtual server to share the same IP address as a node in an associated VLAN. To do this, 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 a self-IP address to the VLAN group, and disabling the virtual server on the relevant VLAN. For more information, see the chapter that describes VLANs and VLAN groups in the ***BIG-IP Network and System Management Guide***.

Note that the BIG-IP system automatically associates a Fast L4 profile with this type of virtual server, as a way to increase the speed at which the virtual server processes Layer 4 requests. For more information on the Fast L4 type of profile, see Chapter 5, *Understanding Profiles*.

- **Forwarding traffic to a specific destination IP address**
  A ***Forwarding (IP)*** virtual server is just like other virtual servers, except that a forwarding virtual server has no pool members to load balance. The virtual server simply forwards the packet directly to the destination IP address specified in the client request. When you use a forwarding virtual server to direct a request to its originally-specified destination IP address, the LTM system adds, tracks, and reaps these connections just as with other virtual servers. You can also view statistics for a forwarding virtual servers.

  Note that the BIG-IP system automatically associates a Fast L4 profile with this type of virtual server, as a way to increase the speed at which the virtual server processes Layer 4 requests. For more information on the Fast L4 type of profile, see Chapter 5, *Understanding Profiles*.

- **Increasing the speed of processing HTTP traffic**
  A ***Performance (HTTP)*** virtual server is a virtual server with which you associate a Fast HTTP profile. Together, the virtual server and profile increase the speed at which the virtual server processes HTTP requests.

- **Increasing the speed of processing Layer 4 traffic**
  A ***Performance (Layer 4)*** virtual server is a virtual server to which the BIG-IP system automatically associates a Fast L4 profile. Together, the virtual server and Fast L4 profile increase the speed at which the virtual server processes Layer 4 requests. For more information on the Fast L4 type of profile, see Chapter 5, *Understanding Profiles*.

When you create a virtual server, you specify the pool or pools that you want to serve as the destination for any traffic coming from that virtual server. You also configure its general properties, some configuration options, and other resources you want to assign to it, such as iRules or session persistence types.

The following sections describe the types of virtual servers you can create, as well as their general properties, configuration options, and resources.

Understanding virtual server types

There are two distinct types of virtual servers that you can create: host virtual servers and network virtual servers.

Host virtual servers

A ***host virtual server*** represents a specific site, such as an Internet web site or an FTP site, and it load balances traffic targeted to content servers that are members of a pool.

The IP address that you assign to a host virtual server should match the IP address that DNS associates with the site's domain name. When the LTM system receives a connection request for that site, the LTM system recognizes that the client's destination IP address matches the IP address of the virtual server, and subsequently forwards the client request to one of the content servers that the virtual server load balances.

Network virtual servers

A ***network virtual server*** is a virtual server whose IP address has no bits set in the host portion of the IP address (that is, the host portion of its IP address is **0**). There are two kinds of network virtual servers: those that direct client traffic based on a range of destination IP addresses, and those that direct client traffic based on specific destination IP addresses that the LTM system does not recognize.

Directing traffic for a range of destination IP addresses

With an IP address whose host bit is set to **0**, a virtual server can direct client connections that are destined for an entire range of IP addresses, rather than for a single destination IP address (as is the case for a host virtual server). Thus, when any client connection targets a destination IP address that is in the network specified by the virtual server IP address, the LTM system can direct that connection to one or more pools associated with the network virtual server.

For example, the virtual server can direct client traffic that is destined for any of the nodes on the **192.168.1.0** network to a specific load balancing pool such as **ingress-firewalls**. Or, a virtual server could direct a web connection destined to any address within the subnet **192.168.1.0/24,** to the pool **default_webservers**.

Directing traffic for transparent devices (wildcard virtual servers)

Besides directing client connections that are destined for a specific network or subnet, a network virtual server can also direct client connections that have a specific destination IP address that the virtual server does not recognize, such as a transparent device. This type of network virtual server is known as a wildcard virtual server.

Wildcard virtual servers are a special type of network virtual server designed to manage network traffic that is targeted to transparent network devices. Examples of transparent devices are firewalls, routers, proxy servers, and cache servers. A wildcard virtual server manages network traffic that has a destination IP address unknown to the LTM system.

Handling unrecognized client IP addresses

A host-type of virtual server typically manages traffic for a specific site, When the LTM system receives a connection request for that site, the LTM 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 load balancing transparent nodes, the LTM system might not recognize a client's destination IP address. The client might be connecting to an IP address on the other side of the firewall, router, or proxy server. In this situation, the LTM system cannot match the client's destination IP address to a virtual server IP address.

Wildcard network virtual servers solve this problem by not translating the incoming IP address at the virtual server level on the LTM system. For example, when the LTM system does not find a specific virtual server match for a client's destination IP address, the LTM system matches the client's destination IP address to a wildcard virtual server, designated by an IP address of **0.0.0.0**. The LTM 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.

Understanding default and port-specific wildcard servers

There are two kinds of wildcard virtual servers that you can create:

- **Default wildcard virtual servers**
  A ***default wildcard virtual server*** is a wildcard virtual server that uses port **0** and handles traffic for all services. 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. For the procedure to create a default wildcard server, see *Creating a wildcard virtual server*, on page 2-7.

- **Port-specific wildcard virtual servers**
  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. 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. For the procedure to create a port-specific wildcard virtual server, see *To create a port-specific wildcard virtual server*, on page 2-8.

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.

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.

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.

In some configurations, you need to set up a wildcard virtual server on one side of the LTM system to load balance connections across transparent devices. You can create another wildcard virtual server on the other side of the LTM system to forward packets to virtual servers receiving connections from the transparent devices and forwarding them to their destination.

Creating and modifying virtual servers

Using the Configuration utility, you can either create a virtual server or modify the settings of an existing virtual server. The following sections contain the procedures for creating and modifying virtual servers. To understand individual virtual server properties and settings, see *Configuring virtual server and virtual address settings*, on page 2-10. For information on viewing existing virtual server configurations, see *Managing virtual servers and virtual addresses*, on page 2-15.

Creating a virtual server

When you create a virtual server, you can create either a host or network virtual server, or a special type of network virtual server called a wildcard virtual server.

Creating a host or network virtual server

You can use the same procedure to create both a host virtual server and a network virtual server. The following procedure creates the most basic host or network virtual server, with all of the default settings. After performing this procedure, you have a load-balancing virtual server that directs traffic to a load balancing pool, using the default settings.

In most cases, creating a basic virtual server satisfies your load balancing or forwarding needs. When you create a basic virtual server, most of the settings are hidden, to simplify the creation process. If you want to adjust other settings beyond the basic ones, you can view and configure more advanced settings. For information on configuring specific settings, see *Configuring virtual server and virtual address settings*, on page 2-10, or see the online help.

◆ Note

In a redundant-system configuration, you cannot create a virtual server for unit 2 unless you have first created a virtual server for unit 1.

To create a host or network virtual server

1. On the Main tab, expand **Local Traffic**.
2. Click **Virtual Servers**.
   The Virtual Servers screen displays.
3. On the upper right portion of the screen, click the **Create** button.
   The New Virtual Server screen opens.
4. Configure all required settings.
   If you are creating a network virtual server, you must set the host bit of the IP address to **0**.

5. Retain or change the values for any optional settings.
6. Click **Finished**

◆ Note

If a virtual server is to have 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 more information, see the online help pertaining to VLANs.

Creating a wildcard virtual server

A wildcard virtual server is a special type of network virtual server. 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 (default or port-specific).
- Finally, you must ensure that port translation is disabled for each virtual server. Port translation is disabled by default.

The following procedures describe how to perform these tasks using the Configuration utility.

To create a pool of transparent devices

To create a pool of transparent devices, display the Pools screen and click the **Create** button. For more information, see Chapter 4, *Configuring Load Balancing Pools*.

To create a default wildcard virtual server

1. On the Main tab, expand **Local Traffic**.
2. Click **Virtual Servers**.
   The Virtual Servers screen opens.
3. On the upper right portion of the screen, click the **Create** button.
   The New Virtual Server screen opens.
4. Configure all required settings.
   Remember to type the IP address **0.0.0.0** in the **Destination Address** box, and if you selected a network type of virtual server, to type the netmask **0.0.0.0** in the **Mask** box.
5. Click **Finished**.

To create a port-specific wildcard virtual server

1. On the Main tab, expand **Local Traffic**.
2. Click **Virtual Servers**.
   The Virtual Servers screen opens.
3. On the upper-right portion of the screen, click **Create**.
   The New Virtual Server screen opens.
4. In the **Address** box, type the wildcard IP address **0.0.0.0**.
5. For the **Service Port** setting, 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.
6. For the **Default Pool** setting in the Resources section, select the pool of transparent devices that you want to apply to the virtual server.
7. Click **Finished**.

To turn off port translation for a wildcard virtual server

After you define the wildcard virtual server with a wildcard port, you should verify that port translation is disabled for the virtual server.

1. On the Main tab, expand **Local Traffic**.
2. Click **Virtual Servers**.
   The Virtual Servers screen opens.
3. In the Name column, click the virtual server for which you want to turn off port translation.
   The Virtual Servers screen opens.
4. In the Enable Translation section, verify that the **Port** box is cleared.

Modifying a virtual server

You can easily modify the settings of an existing virtual server, using the Configuration utility. For information on virtual settings, see *Configuring virtual server and virtual address settings*, on page 2-10 or the online help.

To modify an existing virtual server

1. On the Main tab, expand **Local Traffic**.
2. Click **Virtual Servers**.
   The Virtual Servers screen opens.

3. In the Name column, click the name of the virtual server that you want to modify.
   The properties screen opens for that virtual server.
4. From the Configuration list, select **Advanced**.
5. In the Configuration section, retain or modify any virtual server settings.
6. Click **Update**.
7. On the menu bar, click **Resources.**
   The screen displays additional settings for the selected virtual server.
8. In the Load Balancing section, retain or modify any virtual server resources.
9. Click **Update**.
10. If you want to modify the set of iRules assigned to the virtual server, click the **Manage** button, and use the left or right arrows (**<<** or **>>**) to enable or disable any existing iRules.
11. Click **Finished**.

Configuring virtual server and virtual address settings

A virtual server and its virtual server address have a number of properties and settings that you can configure to affect the way that a virtual server manages traffic. You can also assign certain resources to a virtual server, such as a load balancing pool and a persistence profile. Together, these properties, settings, and resources represent the definition of a virtual server or its address, and most have default values. When you create a virtual server, you can either retain the default values or adjust them to suit your needs.

The following sections list and describe all properties, configuration settings, and resources that define virtual servers and virtual addresses.

For information on how to create virtual server, see *Creating a host or network virtual server*, on page 2-6.

Configuring virtual server properties, settings, and resources

In the Configuration utility, virtual server settings are grouped into three categories: General properties, configuration settings (basic and advanced), and resources (basic and advanced). The following sections describe the settings that these three categories contain.

General properties

When you create a virtual server, you define some general properties. Table 2.1 lists and describes these general properties.

Property	Description	Default Value
Name	A unique name that you assign to the virtual server. This property is required.	No default value
Destination Type	The type of virtual server you want to create and its IP address. If the type you select is **network**, then this property also includes the mask for the IP address. For more information on virtual server types, see *Understanding virtual server types*, on page 2-3. This property is required.	**Host**
Destination Address	The IP address of the virtual server.	No default value

Table 2.1 *General properties of a virtual server*

Property	Description	Default Value
Destination Mask	The netmask for a network virtual server. This property applies to a network virtual server only, and is required. The netmask clarifies whether the host bit is an actual zero or a wildcard representation.	No default value
Service Port	A service name or port number for which you want to direct traffic. This property is required.	No default value
State	The state of the virtual server, that is, **enabled** or **disabled**. As an option, you can enable or disable a virtual server for a specific VLAN. Note that when you disable a virtual server, the virtual server no longer accepts new connection requests. However, it allows current connections to finish processing before going to a **down** state. **Note:** If no VLAN is specified, then the **enabled** or **disabled** setting applies to all VLANs.	**Enabled**

Table 2.1 *General properties of a virtual server*

Configuration settings

When creating a virtual server, you can configure a number of settings. Table 2.2 lists and describes these virtual server configuration settings. Because all of these settings have default values, you are not required to change these settings.

Setting	Description	Default Value
Type	The type of virtual server configuration. Choices are: **Standard**, **IP Forwarding (IP)**, **Forwarding (Layer 2)**, **Performance (HTTP)**, **Performance (Layer 4)**, and **Reject**. For more information, see *Introducing virtual servers*, on page 2-1. Note that if set to **Reject**, this setting causes the BIG-IP system to reject any traffic destined for the virtual server IP address.	**Standard**
Protocol	The network protocol name for which you want the virtual server to direct traffic. Sample protocol names are **TCP** and **UDP**. 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 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. Note that this setting is disabled when creating a Performance (HTTP) type of virtual server.	**TCP**
Protocol Profile (Client)	A setting that designates the selected profile as a client-side profile. Applies to TCP and UDP connections only. When creating a Performance (HTTP) type of virtual server, this value is set to **fasthttp**, and you cannot change it. Similarly, when creating a Performance (Layer 4) type of virtual server, this value is set to **fastl4**, and you cannot change it.	**TCP**

Table 2.2 *Configuration settings for a virtual server*

Setting	Description	Default Value
Protocol Profile (Server)	A setting that designates the selected profile as a server-side profile. Applies to TCP and UDP connections only. Note that this setting does not appear when creating a Performance (HTTP) or Performance (Layer 4) type of virtual server.	**(Use Client Profile)**
OneConnect Profile	The name of an existing OneConnect™ profile for managing connection persistence. Note that this setting does not appear when creating a Performance (HTTP) or Performance (Layer 4) type of virtual server.	**oneconnect**
HTTP Profile	The name of an existing HTTP profile for managing HTTP traffic. Note that this setting does not appear when creating a Performance (HTTP) or Performance (Layer 4) type of virtual server.	**None**
FTP Profile	The name of an existing FTP profile for managing FTP traffic. Note that this setting does not appear when creating a Performance (HTTP) or Performance (Layer 4) type of virtual server.	**None**
Client SSL Profile	The name of an existing SSL profile for managing client-side SSL traffic. Note that this setting does not appear when creating a Performance (HTTP) or Performance (Layer 4) type of virtual server.	**None**
Server SSL Profile	The name of an existing SSL profile for managing server-side SSL traffic. Note that this setting does not appear when creating a Performance (HTTP) or Performance (Layer 4) type of virtual server.	**None**
Authentication Profile	The name of an existing authentication profile for managing an authentication mechanism. Examples are a remote LDAP or RADIUS server. Note that this setting does not appear when creating a Performance (HTTP) or Performance (Layer 4) type of virtual server.	**None**
Stream Profile	The name of an existing Stream profile for managing Real-Time Streaming Protocol (RTSP) traffic. Note that this setting does not appear when creating a Performance (HTTP) or Performance (Layer 4) type of virtual server.	**None**
Statistics Profile	The name of a statistics profile.	**stats**
VLAN Traffic	The names of VLANS for which the virtual server is enabled or disabled.	**ALL VLANS**
Rate Class	The name of an existing rate class, used for enforcing a throughput policy for incoming network traffic.	**None**
Connection Limit	The maximum number of concurrent connections allowed for the virtual server. Setting this to **0** turns off connection limits.	**0**
Address Translation	A setting to enable or disable address translation on an LTM system. This option is useful when the LTM 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.	**Enabled**
Port Translation	A setting to enable or disable port translation on an LTM system. 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.	**Enabled**
SNAT Pool	The name of an existing SNAT pool, used for implementing selective and intelligent SNATs.	**None**

Table 2.2 *Configuration settings for a virtual server*

Setting	Description	Default Value
Clone Pool (Client)	A feature that causes the virtual server to replicate client-side traffic (that is, prior to address translation), to a member of the specified clone pool. This feature is used for intrusion detection. You can also configure the **Clone Pool (Server)** setting.	**None**
Clone Pool (Server)	A feature that causes the virtual server to replicate server-side traffic (that is, after address translation), to a member of the specified clone pool. This feature is used for intrusion detection.You can also configure the **Clone Pool (Client)** setting.	**None**
Last Hop Pool	A setting that directs reply traffic to the last hop router using a last hop pool. This overrides the **auto_lasthop** setting. In cases where you have more than one router sending connections to the LTM 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.) Before configuring a last hop pool, you must first create a pool containing the router inside addresses.	**None**

Table 2.2 *Configuration settings for a virtual server*

Resources

If you have created a load balancing virtual server, you must assign a default load balancing pool to it. If you plan on using an iRule to direct certain kinds of traffic, you must also assign the iRule to the virtual server, as well as to any pools that are specified within the iRule. These pool and iRule assignments are known as ***resources***.

Table 2.3 lists and describes the specific resources that you can assign to a load balancing virtual server.

Resource	Description	Default Value
Default Pool	The pool name that you would like the virtual server to use as the default pool. A load balancing virtual server sends traffic to this pool automatically, unless an iRule directs the server to send the traffic to another pool instead.	No default value

Table 2.3 *Resources assigned to a load balancing virtual server*

Resource	Description	Default Value
Default Persistence Profile	The type of persistence that you want the LTM system to use.	**None**
Fallback Persistence Profile	The type of persistence that the LTM system should use if it cannot use the specified default persistence.	**None**

iRules	A list of existing iRules that you want the virtual server to use when load balancing its connections. Note that for all iRules that you select, you must configure a corresponding profile on the virtual server. For example, if you are specifying an iRule that includes HTTP commands, you must configure a default or custom HTTP profile on the virtual server. Similarly, if you are implementing an authentication iRule, you must configure a default or custom authentication profile.
	
	Once you assign a pool and an iRule to a virtual server, then if all nodes in the pool become unavailable, the virtual server becomes unable to forward the types of requests articulated in the iRule. For example, the following iRule **my_ruleA** causes the virtual server to forward requests to pool **bigip1** when the URI ends with the string **fuchu**. If all nodes in pool **bigip1** are down, the virtual server is unable to forward those requests.
	
	rule my_ruleA {
	if { [HTTP:uri] ends_with "fuchu" } {
	pool bigip1
	}
	else {
	pool bigip2
	}
	}
	
	If the iRule you want to implement does not appear in the **iRules** list, the iRule does not exist and you must first create it. If the **iRules** setting does not appear on the New Virtual Server screen, check your licensing.
		No default value

Table 2.3 *Resources assigned to a load balancing virtual server*

Configuring virtual address properties and settings

The Configuration utility displays virtual address properties and settings. Table 2.4 lists and describes the general properties and configuration settings of a virtual address.

Property	Description	Default Value
Address	The IP address of the virtual server, not including the service.	No default value
State	The state of the virtual address, that is, **enabled** or **disabled**.	**Enabled**
Connection limit	The number of concurrent connections that the LTM system allows on this virtual address.	**0**
ARP	A setting that enables or disables ARP requests.	**Enabled**

Table 2.4 *General properties and configuration settings of a virtual address*

Managing virtual servers and virtual addresses

When generally managing virtual servers and virtual addresses, you typically need to view existing virtual server or virtual address configurations. Occasionally, too, you might need to delete a virtual server.

When working with virtual servers that you have created, you can:

- View a virtual server configuration
- View a virtual address configuration
- Delete a virtual server

Viewing a virtual server configuration

Occassionally, you might want to determine whether you need to adjust virtual server settings, or create new virtual servers. When you view a virtual server configuration, you can view:

- A list of virtual servers
- Virtual server properties and settings
- Virtual server resources
- Virtual server statistics

Viewing a list of virtual servers

You can view a list of existing virtual servers that you have created. When you display the list of virtual servers, the Configuration utility displays the following information about each virtual server:

- Status
- Virtual server name
- Destination (virtual address)
- Service Port
- Protocol
- Resource type (load balancing, forwarding, or L2 forwarding)

To view a list of virtual servers

1. On the Main tab, expand **Local Traffic**.
2. Click **Virtual Servers**.
   A list of all existing virtual servers appears.

Viewing virtual server properties and settings

You can view virtual server properties, such as the profile types that are assigned to the virtual server.

To view virtual server properties

1. On the Main tab, expand **Local Traffic**.
2. Click **Virtual Servers**.
   A list of all existing virtual servers appears.
3. In the Name column, click a virtual server name.
   This displays the properties for that virtual server.

Viewing virtual server resources

You can view the default pool, default persistence profile, and fallback persistence profile that are assigned as resources to the virtual server. You can also view any iRules associated with the virtual server. The following procedure shows how to view these resources.

To view virtual server resources

1. On the Main tab, expand **Local Traffic**.
2. Click **Virtual Servers**.
   A list of all existing virtual servers appears.
3. Click a virtual server name.
   This displays the properties for that virtual server.
4. On the menu bar, click **Resources**.
   This displays the resources assigned to the virtual server.

Viewing virtual server statistics

Using the Configuration utility, you can view statistics for any existing virtual servers.

To view statistics for a virtual server

1. On the Main tab, expand **Local Traffic**.
2. Click **Virtual Servers**.
   A list of all existing virtual servers appears.
3. In the Name column, click the name of a virtual server.
4. From the Statistics menu, choose Virtual Server.
   This displays the statistics for the virtual server.

Viewing a virtual address configuration

Occassionally, you might want to view virtual address settings, to determine whether you need to adjust them. In working with virtual address configurations, you can view:

- A list of virtual addresses
- Virtual address properties
- Virtual address statistics

Viewing a list of virtual addresses

You can view a list of existing virtual addresses that you have created, and adjust any of their settings. When you display the list of virtual addresses, the Configuration utility also displays the state of that address (**enabled** or **disabled**).

To view a list of virtual addresses

1. On the Main tab, expand **Local Traffic**.
2. Click **Virtual Servers**.
   By default, a list of all existing virtual servers appears.
3. On the menu bar, click **Virtual Address List**.
   A list of all existing virtual addresses appears.

Viewing virtual address properties

The following procedure shows how to view virtual address properties.

To view or adjust virtual address properties

1. Display the list of existing virtual addresses, using the previous procedure.
2. Click a virtual address.
   This displays the properties for that virtual address.

Viewing virtual address statistics

Using the Configuration utility, you can view statistics for any existing virtual addresses.

To view statistics for a virtual address

1. On the Main tab, expand **Local Traffic**.
2. Click **Virtual Servers**.
   By default, a list of all existing virtual servers appears.

3. On the menu bar, click Virtual Address List.
   A list of all existing virtual addresses appears.
4. From the Statistics menu, choose **Virtual Address**.
   This displays statistics for the virtual address.

Deleting a virtual server

You can permanently delete a virtual server and its virtual address from a configuration.

To delete a virtual server

1. On the Main tab, expand **Local Traffic**.
2. Click **Virtual Servers**.
   This displays a list of existing virtual servers.
3. Check the **Select** box to the left of any virtual server that you want to delete.
4. Click **Delete**.
   This displays the Delete Confirmation screen.
5. Click **Delete**.
   This removes the virtual server.

3

Configuring Nodes

- Introducing nodes
- Creating and modifying nodes
- Configuring node settings
- Managing nodes

Introducing nodes

Nodes are the network devices to which a BIG-IP® local traffic management (LTM) system passes traffic. You can explicitly create a node, or you can instruct the LTM system to automatically create one when you add a pool member to a load balancing pool.

The difference between a node and a pool member is that a node is designated by the device's IP address only (**10.10.10.10**), while designation of a pool member includes an IP address and a service (such as **10.10.10:80**).

A primary feature of nodes is their association with health monitors. Like pool members, nodes can be associated with health monitors as a way to determine server status. However, a health monitor for a pool member reports the status of a service running on the device, whereas a health monitor associated with a node reports status of the device itself.

For example, if an ICMP health monitor is associated with node **10.10.10.10**, which corresponds to pool member **10.10.10.10:80**, and the monitor reports the node as being in a **down** state, then the monitor also reports the pool member as being **down**. Conversely, if the monitor reports the node as being in an **up** state, then the monitor reports the pool member as being either **up** or **down**, depending on the status of the service running on it.

You create a node using the Configuration utility, and then adjust the settings as needed. Using the same utility, you can also display information about nodes, enable and disable nodes, and delete nodes.

Creating and modifying nodes

Nodes are the basis for creating a load balancing pool. For any server that you want to be part of a load balancing pool, you must first create a node, that is, designate that server as a node. After designating the server as node, you can add the node to a pool as a pool member. You can also associate a health monitor with the node, to report the status of that server. For information on adding nodes to load balancing pools, see Chapter 4, *Configuring Load Balancing Pools*.

You use the Configuration utility to create a node. When you create a node, the LTM system automatically assigns a group of default settings to that node. You can retain these default settings or modify them. You can also modify the settings at a later time, after you have created the node. For information on these settings, see either *Configuring node settings*, on page 3-3, or the online help.

It is helpful to understand that the LTM system designates some settings as basic and others as advanced. If you decide to modify some of the default settings when you create the node, be sure to select the **Advanced** option on the screen to view all configurable settings. For more information on basic and advanced settings, see Chapter 1, *Introducing Local Traffic Management*.

To create a node

1. On the Main tab, expand **Local Traffic**.
2. Click **Nodes**.
   The Nodes screen opens.
3. In the upper-right corner of the screen, click **Create**.
   The New Node screen opens.
4. For the **Address** setting, type the IP address of the node.
5. Specify, retain, or change each of the other settings.
6. Click **Finished**.

To modify an existing node

1. On the Main tab, expand **Local Traffic**.
2. Click **Nodes**.
   The Nodes screen opens.
3. In the Address column, click an address.
   This displays the settings for that node.
4. Retain or modify any node settings.
5. Click **Update**.

Configuring node settings

You can configure node settings to tailor nodes to your specific needs. For those settings that have default values, you can retain those default settings or modify them. Also, you can modify settings either when you create the node, or at any time after you have created it.

Table 3.1 lists these configurable settings and their default values. Following this table are descriptions of specific settings.

Node settings	Description	Default Value
Address	Specifies the IP address of the node. This setting is required.	No default value
Name	Specifies the name of the node.	No default value
Health Monitors	Associates a health or performance monitor with a node.	No default value
Select Monitors	Specifies the monitors that the LTM system is to associate with the node.	No default value
Availability Requirement	Specifies the minimum number of health monitors that must report a node as being available to receive traffic before the LTM system reports that node as being in an **up** state.	**All**
Ratio	Specifies the ratio weight you want to assign to the node.	**1**
Connection Limit	Specifies the maximum number of concurrent connections allowed on a node.	**0**

Table 3.1 *Node configuration settings*

Before configuring a node, it is helpful to have a description of certain node settings that you might want to change.

Specifying an address for a node

For each node that you configure, you must specify an IP address. An example of a node IP address is **10.10.10.10**. This is the only required setting.

Specifying a node name

For each node that you configure, you can give it a unique node name, such as **Node_1**. Node names are case-sensitive and may contain letters, numbers, and underscores (_) only. Reserved keywords are not allowed.

Specifying monitor associations

Using the LTM system, you can monitor the health of performance of your nodes by associating health or performance monitors with those nodes. This is similar to associating a monitor with a load balancing pool, except that in the case of nodes, you are monitoring the IP address, whereas with pools, you are monitoring the services that are active on the pool members.

The LTM system contains many different pre-configured monitors that you can associate with nodes, depending on the type of traffic you want to monitor. You can also create your own custom monitors and associate them with nodes. The only pre-configured monitors that are not available for associating with nodes are monitors that are specifically designed to monitor pools or pool members rather than nodes.

There are two ways that you can associate a monitor with a node: by assigning the same monitor to multiple nodes at the same time, or by explicitly associating a monitor with each node as you create it.

◆ Note

*If you use the **bigpipe** utility commands **node** and **save** to assign the same monitor to multiple nodes (for example, **bigpipe node all monitor icmp** and **bigpipe save**), the LTM system creates a separate monitor-node entry in the **bigip.conf** file for each node.*

For more information about health and performance monitors, see Chapter 10, *Configuring Monitors*.

Creating a default monitor association

You can configure the LTM system to automatically associate one or more monitor types with every node that you create. In this case, you choose the monitors before you create your nodes, and then by default, the LTM system associates those monitors with every node that you create. This eliminates the task of having to explicitly associate monitors with each node that you create.

You designate the default monitor for your nodes by navigating to the Nodes page and using the Default Monitor menu. Then when you create your node, you can set the value of the **Health Monitors** setting to **Node Default**.

◆ Note

*If you want to assign a default monitor to an existing node, you can use the same **Default Monitor** menu to do this.*

Explicitly associating monitors with a node

Instead of configuring the LTM system to automatically associate one or more monitors with every node that you create, you can associate specific monitors with a node when you create that node.

You associate a monitor to a specific node (instead of all nodes as in the case of default monitor associations) by setting the value of the **Health Monitors** setting to **Node Specific** when you create a node or modify a node's settings. The Configuration utility then allows you to choose from a list of monitors that are available for associating with that node.

Specifying the availability requirement

By configuring the **Availability Requirement** setting, you can specify the minimum number of health monitors that must report a node as being available to receive traffic before the LTM system reports that node as being in an **up** state. Acceptable values are **All** or a number that you specify. If you choose the value A**t Least**, you then specify a number.

Specifying a ratio weight

The **Ratio** setting specifies a ratio weight for the node. The default setting is **1**. For information on ratio weights, see Chapter 4, *Configuring Load Balancing Pools*.

Setting a connection limit

Using the **Connection Limit** setting, you can specify the maximum number of concurrent connections allowed for a node. Note that the default value of **0** (zero) means that there is no limit to the number of concurrent connections that the node can receive.

Managing nodes

After you have created your nodes and configured their settings to suit your needs, you might want to perform some additional management tasks. You can manage existing nodes in these ways:

- Viewing existing nodes
- Enabling or disabling existing nodes
- Deleting existing nodes
- Disabling monitor associations
- Displaying node status

Viewing existing nodes

You can use the following procedure to view nodes that you have created.

To view existing nodes

1. On the Main tab, expand **Local Traffic**.
2. Click **Nodes**.
   This displays a list of existing nodes.
3. In the Address column, click the address of the node you want to view.
   This displays the settings for that node.

Enabling and disabling a node

A node must be enabled in order to accept traffic. When a node is disabled, the LTM system allows existing connections to time out or end normally. In this case, the node can accept new connections only if the connections belong to an existing persistence session. (In this way a disabled node differs from a node that is set to **down**. The **down** node allows existing connections to time out, but accepts no new connections.)

To enable or disable a node

1. On the Main tab, expand **Local Traffic**.
2. Click **Nodes**.
   This displays a list of existing nodes.
3. Locate the address of the node you want to enable or disable.
4. In the column to the left, check the Select box.
5. At the bottom of the screen, click the **Enable** or **Disable** button.

Deleting a node

If you are no longer using a node in a pool, you can delete the node.

To delete a node

1. On the Main tab, expand **Local Traffic**.
2. Click **Nodes**.
   This displays a list of existing nodes.
3. Locate the address of the node you want to enable or disable.
4. In the column to the left, check the Select box.
5. On the bottom of the screen, click **Delete**.
   A confirmation screen appears.
6. Click **Delete**.

Removing monitor associations

Using the Configuration utility, you can remove a monitor that is explicitly associated with a specific node. When removing a monitor associated with a specific node, you can either remove the monitor association altogether, or change it so that only the default monitor is associated with the node.

Alternatively, you can also remove any default monitors, that is, monitors that the LTM system automatically associates with any node that you create.

For more information on monitor associations, see *Specifying monitor associations*, on page 3-4.

To remove an explicit monitor association for a node

1. On the Main tab, expand **Local Traffic**.
2. Click **Nodes**.
   This displays a list of existing nodes.
3. Click the address of the node you want to manage.
4. In the Configuration section of the screen, locate the Health Monitors setting.
5. Select **Node Default** or **None**.
6. Click **Update**.

To remove a default monitor

1. On the Main tab, expand **Local Traffic**.
2. Click **Nodes**.
   This displays a list of existing nodes.
3. Click the Default Monitor menu.
4. Using the right arrows (**>>**), move any active monitors from the **Active** box to the **Available** box.
5. Click **Update**.

Displaying node status

The Nodes screen automatically displays the status of all existing nodes. Available status values are:

- All
- Available
- Unavailable
- Offline
- Unknown

4

Configuring Load Balancing Pools

- Introducing load balancing pools
- Creating and modifying load balancing pools
- Configuring pool settings
- Configuring pool member settings
- Managing pools and pool members

Introducing load balancing pools

In a typical client-server scenario, a client request goes to the destination IP address specified in the header of the request. For sites with a large amount of incoming traffic, the destination server can quickly become overloaded as it tries to service a large number of requests. To solve this problem, the BIG-IP® local traffic management (LTM) system distributes client requests to multiple servers instead of to the specified destination IP address only. You configure the LTM system to do this when you create a load balancing pool.

What is a load balancing pool?

A ***load balancing pool*** is a set of devices, such as web servers, that you group together to receive and process traffic. Instead of sending client traffic to the destination IP address specified in the client request, the LTM system sends the request to any of the servers that are members of that pool.

When you create a pool, you assign servers (known as pool members) to the pool, and then associate the pool with a virtual server in the LTM system. The LTM system then directs traffic coming into the virtual server to a member of that pool. An individual server can belong to one or multiple pools, depending on how you want to manage your network traffic.

The specific pool member to which the LTM system chooses to send the request is determined by the load balancing method that you have assigned to that pool. A ***load balancing method*** is an algorithm that the LTM system uses to select a pool member for processing a request. For example, the default load balancing method is ***Round Robin***, which causes the LTM system to send each incoming request to the next available member of the pool, thereby distributing requests evenly across the servers in the pool. For a complete list of load balancing methods, see *Specifying the load balancing method*, on page 4-10.

Features of a load balancing pool

You can configure the LTM system to perform a number of different operations for a pool. You can:

- Associate health monitors with pools and pool members
- Enable or disable SNAT connections
- Rebind a connection to a different pool member if the originally-targeted pool member becomes unavailable
- Set the Quality of Service or Type of Service level within a packet
- Specify a load balancing algorithm for a pool
- Assign pool members to priority groups within a pool

Creating and modifying load balancing pools

You use the Configuration utility to create a load balancing pool, or to modify a pool and its members. When you create a pool, the LTM system automatically assigns a group of default settings to that pool and its members. You can retain these default settings or modify them. Also, you can modify the settings at a later time, after you have created the pool.

It is helpful to understand that the LTM system designates some settings as basic and others as advanced. If you decide to modify some of the default settings when you create the pool, be sure to select the **Advanced** option on the screen to view all configurable settings. For more information on basic and advanced settings in general, see Chapter 1, *Introducing Local Traffic Management*.

Creating and implementing a load balancing pool

Creating and implementing a load balancing pool is a two-task process:

- First, you must create the pool.
- Second, you must associate the pool with a virtual server.

To create a load balancing pool

1. On the Main tab, expand **Local Traffic**.
2. Click **Pools**.
   The Pools screen opens.
3. In the upper-right corner of the screen, click **Create**.
   The New Pool screen opens.
4. From the Configuration list, select **Advanced**.
5. For the **Name** setting, type a name for the pool.
6. Specify, retain, or change each of the other settings.
   For information on pool settings, see *Configuring pool settings*, on page 4-5, or refer to the online help for this screen.
7. Click **Finished**.

To implement a load balancing pool

1. On the Main tab, click **Virtual Servers**.
   The Virtual Servers screen opens.
2. Click the name of the appropriate virtual server.
   This displays the settings for that virtual server.
3. On the menu bar, click Resources.
4. In the **Default Pool** list, select the name of your newly-created pool.
5. Click **Update**.

Modifying a load balancing pool

You can modify any settings configured for an existing pool, including the load balancing method. For information on pool settings, see *Configuring pool settings*, on page 4-5, or see the online help. For information on adding members to an existing pool, see *Modifying pool membership*, following.

To modify pool settings

1. On the Main tab, expand **Local Traffic**.
2. Click **Pools**.
   The Pools screen opens.
3. Click the name of an existing pool.
   This displays the existing settings for that pool.
4. From the Configuration list, select **Advanced**.
   This displays the pool settings.
5. Modify or retain all settings.
6. Click **Update**.
7. If you want to modify the load balancing method or enable or disable priority group activation, locate the menu bar and click Members.
8. Modify or retain the **Load Balancing Method** and **Priority Group Activation** settings.
9. Click **Update**.

Modifying pool membership

For an existing load balancing pool, you can either modify existing pool members or add new members to the pool.

Modifying existing pool members

When modifying settings for members of a pool, you can:

- Enable or disable pool members
- Remove members from the pool
- Modify the values of pool member settings

To modify existing pool members

1. On the Main tab, expand **Local Traffic**.
2. Click **Pools**.
   The Pools screen opens.
3. In the Members column, click the number shown.
   This lists the existing members of the pool.

4. Locate the Current Members section of the screen.
5. Modify a pool member:
   a) If you want to enable or disable a pool member, or remove a member from the pool, click the box to the left of a member address. Then click **Enable**, **Disable**, or **Remove**.
   b) If you want to modify the settings for a pool member, click an address and retain or modify pool member settings as needed. For information on pool member settings, see *Configuring pool member settings*, on page 4-14.
6. Click **Update**.

Adding members to an existing load balancing pool

Not only can you specify pool members at the time that you create a pool, you can add pool members later on. When adding a pool member to an existing pool (as opposed to specifying a pool member during pool creation), you can configure a number of settings for that pool member. The only settings that you must explicitly specify are the **Address** and **Service Port** settings. All other settings have default values that you can either retain or adjust, depending on your needs.

◆ Note

*If you specify a pool member at the time that you create a pool, you do not see these settings; instead, the LTM system simply assigns default values. However, you can adjust the settings later by modifying the pool member properties. For more information, see **Managing pools and pool members**, on page 4-18.*

To add members to a load balancing pool

1. On the Main tab, expand **Local Traffic**.
2. Click **Pools**.
   The Pools screen opens.
3. In the Members column, click the number shown.
   This lists the existing members of the pool.
4. In the right side of the screen, click **Add**.
   The New Pool Members screen opens.
5. In the **Address** box, select **New Address** and type an IP address, or select **Node List** and select an IP address.
6. In the **Service Port** box, type a port number or select a service from the list.
7. Retain or configure all other settings. For information on pool member settings, see *Configuring pool member settings*, on page 4-14.
8. Click **Finished**.

Configuring pool settings

You can configure pool settings to tailor pools to your specific needs. For those settings that have default values, you can retain those default settings or modify them. Also, you can modify any settings either when you create the pool, or at any time after you have created it. For information on how to use the Configuration utility to configure these settings, see *Creating and modifying load balancing pools*, on page 4-2.

Table 4.1 lists the settings that you can configure for a pool, followed by a description of each setting.

Pool Setting	Description	Default Value
Name	You can specify the user-supplied name of the pool. Specifying a name for a pool is required.	No default value
Health Monitors	You can associate a health or performance monitor with an entire pool, rather than with individual pool members only. This eases the task of configuring health and performance monitoring for multiple web servers.	No default value
Availability Requirement	You can specify the number of monitors that must report a pool member as being available before that member is defined as being in an **up** state.	**All**
Allow SNAT	You can configure a pool so that SNATs are automatically enabled or disabled for any connections using that pool.	**Yes**
Allow NAT	You can configure a pool so that NATs are automatically enabled or disabled for any connections using that pool.	**Yes**
Action on Service Down	If this setting is enabled and the target pool member goes down, the LTM system tries to choose another pool member and rebind the client connection to a new server connection. Possible values are **None**, **Reject**, **Drop**, and **Reselect**.	**None**
Slow Ramp Time	This option causes the BIG-IP system to send a less-than-normal amount of traffic to a newly-enabled pool member for the specified amount of time.	**0**
Link QoS	You can configure a pool to set a specific Quality of Service (QoS) level within a packet, based on the targeted pool.	**0**
IP ToS	You can configure a pool to set a specific Type of Service (ToS) level within a packet, based on the targeted pool.	**0**

Table 4.1 *Settings for a load balancing pool*

Pool Setting	Description	Default Value
Load Balancing Method	You can use the default load balancing method, or you can define another load balancing method, and you can configure priority-based member activation. Different pools can be configured with different load balancing methods.	**Round Robin**
Priority Group Activation	You can assign pool members to priority groups within the pool.	**Disabled**
New Members	For each pool that you create, you must specify the servers that are to be members of that pool. Pool members must be specified by their IP addresses. For each pool member, you can also assign a service port, a ratio weight, and a priority group.	No default value

Table 4.1 *Settings for a load balancing pool*

Before configuring a pool, it is helpful to have a description of certain pool settings that you might want to change.

Specifying a pool name

The most basic setting 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.

Associating health monitors with a pool

Monitors are a key feature of the LTM system. Monitors help to ensure that a server is in an **up** state and able to receive traffic. When you want to associate a monitor with an entire pool of servers, you do not need to explicitly associate that monitor with each individual server. Instead, you can simply use the pool setting **Health Monitors** to assign the monitor to the pool itself. The LTM system then automatically monitors each member of the pool.

The LTM system contains many different pre-configured monitors that you can associate with pools, depending on the type of traffic you want to monitor. You can also create your own custom monitors and associate them with pools. The only monitor types that are not available for associating with pools are monitors that are specifically designed to monitor nodes and not pools or pool members. These monitor types are:

- ICMP
- TCP Echo
- Real Server
- SNMP DCA

- SNMP DCA Base
- WMI

With the LTM system, you can configure your monitor associations in many useful ways:

- You can associate a monitor with an entire pool instead of an individual server. In this case, the LTM system automatically associates that monitor with all pool members, including those that you add later. Similarly, when you remove a member from a pool, the LTM system no longer monitors that server.
- When a server that is designated as a pool member allows multiple processes to exist on the same IP address and port, you can check the health or status of each process. To do this, you can add the server to multiple pools, and then within each pool, associate a monitor with the that server. The monitor you associate with each server checks the health or performance of the process running on that server.
- When associating a monitor with an entire pool, you can exclude an individual pool member from being associated with that monitor. In this case, you can associate a different monitor for that particular pool member, or you can exclude that pool member from health monitoring altogether. For example, you can associate pool members A, B, and D with the **http** monitor, while you associate pool member C with the **https** monitor.
- You can associate multiple monitors with the same pool. For instance, you can associate both the **http** and **https** monitors with the same pool.

For detailed information on health and performance monitors, see Chapter 10, *Configuring Monitors*.

Specifying the availability requirements

This setting specifies a minimum number of health monitors. Before the LTM system can report the pool member as being in an **up** state, this number of monitors, at a minimum, must report a pool member as being available to receive traffic.

To configure this setting, type a number in the **Availability Requirement** box.

Allowing SNATs and NATs

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. You do this by configuring the **Allow SNAT** and **Allow NAT** settings. By default, these settings are enabled. You can change this setting on an existing pool by displaying the Properties screen for that pool.

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 disable the SNAT or NAT for that pool.

For general information on SNATs and NATs, see Chapter 11, *Configuring SNATs and NATs*.

Specifying action when a service becomes unavailable

The **Action on Service Down** setting specifies the action that you want the LTM system to take when the service on a pool member becomes unavailable. The possible settings are:

- **None** -- The LTM takes no action. This is the default action.
- **Reject** -- The LTM system sends an RST or ICMP message.
- **Drop** -- The LTM system simply cleans up the connection.
- **Reselect** -- The LTM system selects a different node.

To configure this setting, locate the **Action on Service Down** setting and select a value from the list.

Configuring a slow ramp time

When you take a pool member offline, and then bring it back online, the pool member can become overloaded with connection requests, depending on the load balancing mode for the pool. For example, if you use the Least Connections load balancing mode, the system sends all new connections to the newly-enabled pool member (because technically, that member has the least amount of connections).

When you configure the **Slow Ramp Time** setting, the system sends less traffic to the newly-enabled pool member. The amount of traffic is based on the ratio of how long the pool member has been available compared to the slow ramp time, in seconds. Once the pool member has been online for a time greater than the slow ramp time, the pool member receives a full proportion of the incoming traffic.

To configure this setting, locate the **Slow Ramp Time** setting and type a number.

Configuring the Quality of Service (QoS) level

Another setting 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. Essentially, the QoS level specified in a packet enforces a throughput policy for that packet.

As traffic enters the site, the LTM system can set the QoS level on a packet, based on the QoS level that you define for the pool to which the packet is sent. The LTM system can also apply an iRule that sends the traffic to different pools of servers based on the Quality of Service level.

The LTM 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 an iRule, the LTM system can examine incoming traffic to see if it has a particular QoS or ToS tag in the header. The LTM system can then make an iRule-informed load balancing decision based on that tag.

For example, to configure a pool so that a QoS level is set for a packet sent to that pool, you can set the **Client** level to **3** and the **Server** level to **4**. In this case, the QoS level is set to **3** when sending packets to the client and the QoS level is set to **4** when sending packets to the server.

In addition to configuring a pool to set the QoS level on a packet, you can configure an iRule that selects a pool based on the existing QoS value within the packet. For more information, see Chapter 13, *Writing iRules*.

Configuring the Type of Service (ToS) level

Another pool setting is the Type of Service (ToS) level. In addition to the QoS level, the ***ToS*** level is another means by which network equipment can identify and treat traffic differently based on an identifier. As traffic enters the site, the LTM system can set the ToS level on a packet, based on the ToS level that you define for the pool to which the packet is sent. The LTM system can also apply an iRule and send the traffic to different pools of servers based on the ToS level.

The LTM 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 an iRule, the LTM system can examine incoming traffic to see if it has a particular ToS tag in the header. The LTM system can then make an iRule-informed load balancing decision based on that tag.

For example, to configure a pool so that a ToS level is set for a packet sent to that pool, you can set both the **Client** level and the **Server** level to **16**. In this case, the ToS level is set to **16** when sending packets to the client and the ToS level is set to **16** when sending packets to the server.

◆ Note

*If you change the **ToS** level on a pool for a client or a server, 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 an iRule that selects a pool based on the existing ToS value within the packet. For more information, see Chapter 13, *Writing iRules*.

Specifying the load balancing method

Load balancing is an integral part of the LTM system. Configuring load balancing on an LTM system means determining your load balancing scenario, that is, which pool member 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 LTM 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 LTM 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.

Using the default load balancing method

The default load balancing method for the LTM 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 method, you can skip the remainder of this section, and begin configuring other pool settings that you want to add to the basic pool configuration.

Selecting a load balancing method

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

- **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 (member)** and **Ratio (node)**
  The LTM 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. These are static load balancing methods, basing distribution on static user-assigned ratio weights that are proportional to the capacity of the servers. Regarding Ratio load balancing:

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). This distinction is especially important with the Ratio method; with the Ratio (member) method, the actual ratio weight is a member setting in the pool definition, whereas with the Ratio (node) method, the ratio weight is a setting of the node.

The default ratio setting for any node 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 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.

Warning*: If you set the load balancing method to Ratio (node), as opposed to Ratio (Member), you must define a ratio setting for each node.*

- **Dynamic Ratio**
  The Dynamic Ratio method is like the Ratio method 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.

  The Dynamic Ratio method 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, and then install the appropriate performance monitor. For more information, see Appendix A, *Additional Monitor Considerations*.

- **Fastest (node)** and **Fastest (application)**
  The Fastest methods pass a new connection based on the fastest response of all currently active nodes. These methods 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).

- **Least Connections (member)** and **Least Connections (node)**
  The Least Connections methods are relatively simple in that the LTM system passes a new connection to the node that has the least number of current connections. Least Connections methods work best in environments where the servers or other equipment you are load balancing have similar capabilities.

These are dynamic load balancing methods, 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).

- **Observed (member)** and **Observed (node)**
  The Observed methods use a combination of the logic used in the Least Connections and Fastest modes. With the Observed methods, 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. The Observed modes also work well in any environment, but may be particularly useful in environments where node performance varies significantly.

  These are dynamic load balancing methods, 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).

- **Predictive (member)** and **Predictive (node)**
  The Predictive methods also use the ranking methods used by the Observed methods, where nodes are rated according to a combination of the number of current connections and the response time. However, with the Predictive methods, the LTM 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. The Predictive methods work well in any environment.

  The Predictive methods are dynamic load balancing methods, 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).

Specifying 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 pool members available (meaning they are enabled, marked **up**, and have not exceeded their connection limit), the LTM system distributes connections to all members in the highest priority group only, that is, the group designated by the highest priority number. The **Priority Group Activation** value determines the minimum number of members that must remain available for traffic to be confined to that group. If the number of available members in the highest priority group drops below the minimum number, the LTM system also distributes traffic to the next higher priority group, and so on.

```
pool my_pool {
    lb_mode fastest
    min active members 2
    member 10.12.10.7:80 priority 3
    member 10.12.10.8:80 priority 3
    member 10.12.10.9:80 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 10.12.10.1:80 priority 1
    member 10.12.10.2:80 priority 1
    member 10.12.10.3:80 priority 1
    }
```

Figure 4.1 *Sample pool configuration for priority load balancing*

The configuration shown in Figure 4.1 has three priority groups, **3**, **2**, and **1**. Connections are first distributed to all pool members with priority **3** (the highest priority group). If fewer than two priority **3** members are available, traffic is directed to the priority **2** members as well. If both the priority **3** group and the priority **2** group have fewer than two members available, traffic is directed to the priority **1** group. The LTM system continuously monitors the higher priority groups, and each time a higher priority group once again has the minimum number of available members, the LTM system again limits traffic to that group.

Specifying pool members

When you configure this setting, you are specifying the servers (that is, pool members) that will make up the load balancing pool. To specify a pool member, you must specify the server's IP address, and a service port. An optional setting is the ratio weight, applicable when you have selected the load balancing method **Ratio (member)**, **Ratio (node)**, or **Dynamic Ratio**.

Configuring pool member settings

When adding members to a pool, you can configure a number of settings for that pool member. You configure most of these settings after you have created the load balancing pool. The only settings that you must specify during pool creation are the **Address** and **Service Port** settings. All other settings have default values that you can either retain or adjust later, depending on your needs.

For information on adding pool members during pool creation, see *Creating and implementing a load balancing pool*, on page 4-2. For information on adding members to an existing pool, see *Adding members to an existing load balancing pool*, on page 4-4.

Table 4.2 shows the settings that you can configure for a pool member, followed by a description of each setting.

General property	Description	Default Value
Address	Specifies the IP address of a pool member. You can either type an IP address or select one from a list of nodes.	No default value
Service Port	Specifies the service for the pool member.	No default value
Ratio	Specifies the ratio weight that you want to assign to the pool member.	**1**
Priority	You can assign pool members to priority groups within the pool.	**1**
Connection Limit	Specifies the maximum number of concurrent connections allowed for a pool member.	**0**
Health Monitors	Specifies whether you want the pool member to inherit the monitor associated with the pool or to use a different monitor.	**Inherit From Pool**
Select Monitors	Specifies the monitor or monitors that you want to associate with that pool member. This row is used only when the **Health Monitors** row is set to **Member Specific**.	No default value

Table 4.2 *Settings for an individual pool member*

To adjust settings for a pool member after you have added it to a pool, see *To display pool member properties*, on page 4-18.

Before adding pool members, it is helpful to have a description of certain pool member settings that you might want to change.

Specifying an address

When you add a member to a pool, you use the **Address** setting to specify the IP address of that pool member. You can do this either by typing an IP address or by selecting an IP address from the node list. This setting is required.

Specifying a service port

The **Service Port** setting represents either the service name or the port number for the service associated with the pool member. This setting is required.

Specifying a ratio weight for a pool member

When using a ratio-based load balancing method for distributing traffic to servers within a pool, you can use the **Ratio** setting to assign a ratio weight to the server. The ratio weight determines the amount of traffic that the server receives.

The ratio-based load balancing methods are: **Ratio (member)**, **Ratio (node),** and **Dynamic Ratio**. For more information on ratio-based load balancing methods, see *Specifying the load balancing method*, on page 4-10, and Appendix A, *Additional Monitor Considerations*.

Specifying priority-based member activation

The **Priority** setting assigns a priority number to the pool member. Within the pool, traffic is then load balanced according to the priority number assigned to the pool member. Thus, members that are assigned a high priority receive the traffic until the load reaches a certain level, at which time the traffic goes to members assigned to the next lower priority group.

You configure the load level that determines when the LTM system begins directing traffic to members of a lower priority through the pool setting **Priority Group Activation**. For more information, see *Specifying priority-based member activation*, on page 4-13.

Specifying a connection limit

With the **Connection Limit** setting, you can specify the maximum number of concurrent connections allowed for a pool member. Note that the default value of **0** (zero) means that there is no limit to the number of concurrent connections that the pool member can receive.

Selecting an explicit monitor association

Once you have associated a monitor with a pool, the LTM system automatically associates that monitor with every pool member, including those members that you add to the pool later. However, in some cases you might want the monitor for a specific pool member to be different from that assigned to the pool. In this case, you must use the **Health Monitors** setting to specify that you want to explicitly associate a specific monitor with the pool member.

You can also configure this setting to prevent the LTM system from associating any monitor with that pool member.

To explicitly associate a monitor with a pool member, locate the **Health Monitors** setting and select **Member Specific**, which causes the **Select Monitors** setting to appear. Then configure the **Select Monitors** setting as described in the following section.

To ensure that the LTM system associates no monitor with the pool member, set the **Health Monitors** setting to **None**.

Creating an explicit monitor association for a pool member

The LTM system contains many different monitors that you can associate with a pool member, depending on the type of traffic you want to monitor. You can also create your own custom monitors and associate them with pool members. The only monitor types that are not available for associating with pool members are monitors that are specifically designed to monitor nodes and not pools or pool members. These monitor types are:

- ICMP
- TCP Echo
- Real Server
- SNMP DCA
- SNMP DCA Base
- WMI

For detailed information on health and performance monitors, see Chapter 10, *Configuring Monitors*.

To associate a monitor with an individual pool member, you simply display the pool member settings and set the **Health Monitors** setting to **Member Specific**. This displays the **Select Monitors** setting. Select the monitor that you want to associate with the pool member, and using the left arrows (**<<**), move the monitor name to the **Active** box. Clicking **Finished** or **Update** activates the monitor association for that pool member only.

Associating multiple monitors with the same pool member

The LTM system allows you to associate more than one monitor with the same server. Using the Configuration utility, you can:

- **Associate more than one monitor with a member of a single pool**. For example, you can create monitors **http1**, **http2**, and **http3**, where each monitor is configured differently, and associate all three monitors with the same pool member. In this case, the pool member is marked as **down** if any of the checks are unsuccessful.
- **Assign one IP address and service to be a member of multiple pools**. Then, within each pool, you can associate a different monitor with that pool member. For example, suppose you assign the server **10.10.10:80** to three separate pools: **my_pool1**, **my_pool2**, and **my_pool3**. You can then associate all three custom HTTP monitors to that same server (one monitor per pool). The result is that the LTM system uses the **http1** monitor to check the health of server **10.10.10.:80** in pool **my_pool1**, the **http2** monitor to check the health of server **10.10.10.:80** in pool **my_pool2**, and the **http3** monitor to check the health of server **10.10.10:80** in pool **my_pool3**.

You can make multiple-monitor associations either at the time you add the pool member to each pool, or by later modifying a pool member's properties.

Managing pools and pool members

When generally managing pools and pool members, you typically need to view existing pool or pool member configurations. Occasionally, you might need to perform other management tasks as well. Using the Configuration utility, you can:

- Display pool or pool member properties
- Disable monitor associations
- Delete load balancing pools
- View statistics for pools and pool members

Displaying pool or pool member properties

You can display the settings configured for a pool or individual pool member, using the following procedures. For information on modifying pool properties, see *Modifying a load balancing pool*, on page 4-3. for information modifying pool member properties, see *Modifying existing pool members*, on page 4-3.

To display pool properties

1. On the Main tab, expand **Local Traffic**.
2. Click **Pools**.
   The Pools screen opens.
3. Click a pool name.
   This displays the properties for that pool.

To display pool member properties

1. On the Main tab, expand **Local Traffic**.
2. Click **Pools**.
   This displays a list of existing pools.
3. In the Members column, click the number shown.
   This lists the members of that pool.
4. In the Current Members list, click a pool member address.
   This displays the properties for that pool member.

Removing monitor associations

You can remove any existing monitor associations for a pool or pool member.

To remove a monitor from a pool, access the properties page for the pool and change the **Health Monitors** setting by moving the monitor name in the **Active** box to the **Available** box.

To remove an explicit monitor association on an individual pool member, access the properties page for the pool member and change the **Health Monitors** setting to either **Inherit from Pool** or **None**. Selecting **None** excludes the pool member from any monitoring that you have configured on that pool.

Deleting a pool

To delete an existing pool, use the following procedure. For information on removing individual pool members from a pool, see *Modifying existing pool members*, on page 4-3.

Before deleting a pool, you must first remove the pool as a resource from the virtual server.

To delete a pool

1. On the Main tab, expand **Local Traffic**.
2. Click **Pools**.
   This displays a list of existing pools.
3. In the left column next to a pool name, check the Select box.
4. Click **Delete**.
   This displays the Delete Confirmation screen.
5. Click **Delete**.

Viewing pool and pool member statistics

Using the Configuration utility, you can view statistics related to existing pools and pool members.

To view pool and pool member statistics, display the list of existing pools or the list of existing pool members. Then click **Statistics** on the menu bar. This opens the Statistics screen, which shows statistics for all existing pools and their pool members.

The types of statistics shown are:

- Bits (in and out)
- Packets (in and out)
- Connections (current, maximum, and total)

5

Understanding Profiles

- Introducing profiles
- Creating and modifying profiles
- Implementing a profile
- Configuring protocol-type profiles
- Configuring other types of profiles
- Managing profiles
- Using profiles with iRules

Introducing profiles

The BIG-IP® local traffic management (LTM) system can manage application-specific network traffic in a variety of ways, depending on the protocols and services being used. For example, you can configure the LTM system to compress HTTP response data, or you can configure the system to authenticate SSL client certificates before passing requests on to a target server.

For each type of traffic that you want to manage, the LTM system contains configuration tools that you can use to intelligently control the behavior of that traffic. These tools are called profiles. A ***profile*** is a system-supplied configuration tool that enhances your capabilities for managing application-specific traffic. More specifically, a profile is an object that contains user-configurable settings, with default values, for controlling the behavior of a particular type of network traffic, such as HTTP connections. After configuring a profile, you associate the profile with a virtual server. The virtual server then processes traffic according to the values specified in the profile. Using profiles enhances your control over managing network traffic, and makes traffic-management tasks easier and more efficient.

You can associate multiple profiles with a single virtual server. For example, you can associate a TCP profile, an SSL profile, and an HTTP profile with the same virtual server.

Profile types

The LTM system provides several types of profiles. While some profile types correspond to specific protocols, such as HTTP, SSL, and FTP, other profiles pertain to traffic behaviors applicable to multiple protocols. Examples of these are connection persistence profiles and authentication profiles. Table 5.1 lists the available profile types, with descriptions.

Profile Type	Description
Protocol profiles	
Fast L4	Defines the behavior of Layer 4 IP traffic.
Fast HTTP	Improves the speed at which a virtual server processes traffic.
TCP	Defines the behavior of TCP traffic.
UDP	Defines the behavior of UDP traffic.
Services profiles	
HTTP	Defines the behavior of HTTP traffic.
FTP	Defines the behavior of FTP traffic.

Table 5.1 *Available profiles in the LTM system*

Profile Type	Description
SSL profiles	
Client SSL	Defines the behavior of client-side SSL traffic. See also **Persistence Profiles**.
Server SSL	Defines the behavior of server-side SSL traffic. See also **Persistence Profiles**.
Persistence profiles	
Cookie	Implements session persistence using HTTP cookies.
Destination Address	Implements session persistence based on the destination IP address specified in the header of a client request. Also known as sticky persistence.
Hash	Implements session persistence in a way similar to universal persistence, except that the LTM system uses a hash for finding a persistence entry.
MSRDP	Implements session persistence for Microsoft Remote Desktop Protocol sessions.
SIP	Implements session persistence for connections using Session Initiation Protocol Call-ID.
Source Address	Implements session persistence based on the source IP address specified in the header of a client request. Also known as simple persistence.
SSL	Implements session persistence for non-terminated SSL sessions, using the session ID.
Universal	Implements session persistence using the LTM system's Universal Inspection Engine (UIE).
Authentication profiles	
LDAP	Allows the LTM system to authenticate traffic based on authentication data stored on a remote Lightweight Directory Access Protocol (LDAP) server.
RADIUS	Allows the LTM system to authenticate traffic based on authentication data stored on a remote RADIUS server.
TACACS+	Allows the LTM system to authenticate traffic based on authentication data stored on a remote TACACS+ server.
SSL Client Certificate LDAP	Allows the LTM system to control a client's access to server resources based on data stored on a remote LDAP server. Client authorization credentials are based on SSL certificates, as well as defined user groups and roles.
SSL OCSP	Allows the LTM system to check on the revocation status of a client certificate using data stored on a remote Online Certificate Status Protocol (OCSP) server. Client credentials are based on SSL certificates.
Other profiles	
OneConnect	Enables client requests to reuse server-side connections. The ability for the LTM system to reuse server-side connections is known as Connection Pooling™.
Stream	Defines the behavior of Real-Time Streaming Protocol (RTSP) traffic.

Table 5.1 *Available profiles in the LTM system*

Default profiles

The LTM system includes a default profile for each profile type listed in Table 5.1. A ***default profile*** is a system-supplied profile that contains default values for its settings. An example of a default profile is the **http** default profile. You can use a default profile as is, or you can create a custom profile based on the default profile.

You can use a default profile in several ways:

- **You can use a default profile as is.**
  You simply configure your virtual server to reference the default profile.
- **You can modify the default profile settings (not recommended).**
  When you modify a default profile, you lose the original default profile settings. Thus, any custom profiles you create in the future that are based on that default profile inherit the modified settings.
- **You can create a custom profile, based on the default profile (recommended).**
  This allows you to preserve the default profile, and instead configure personalized settings in the custom profile. Custom profiles inherit some of the setting values of a parent profile that you specify. After creating a custom profile, you can configure your virtual server to reference the custom profile instead of the default profile. For more information on custom profiles, see *Custom and parent profiles*, following.

Note

You can modify a default profile, but you cannot create or delete a default profile.

WARNING

Once you modify a default profile, the only way to restore the settings to their original values is to manually configure the profile to specify those values.

Custom and parent profiles

A ***custom profile*** is a profile that you create. When you create a profile, one of the settings that you specify is the **Parent Profile** setting. A ***parent profile*** is a profile from which your custom profile inherits its settings and their default values.

In general, a custom profile automatically inherits the settings and values of its parent profile. Thus, instead of having to configure every individual setting when you create a custom profile, you simply specify a parent profile, which causes the LTM system to automatically assign values to its settings. The values that it assigns are those of the parent profile.

Note

If you do not specify a parent profile when you create a custom profile, the LTM system automatically assigns the corresponding default profile as the parent profile.

Using the default profile as the parent profile

A typical profile that you can specify as a parent profile when you create a custom profile is a default profile. For example, if you create a custom HTTP-type profile called **my_http_profile**, you can specify the default profile **http** as the parent profile. In this case, the LTM system automatically creates the profile **my_http_profile** so that it contains the same settings and default values as the profile **http**. The new child profile thus inherits its settings and values from its parent profile.

An exception to this automatic inheritance feature occurs when you modify the settings of a child profile. In this case, the child profile does not inherit parent values for any of the modified settings. This behavior is designed to prevent the LTM system from overwriting any settings that you have modified in a child profile.

For all settings in a child profile that you do *not* modify, however, the child profile continues to inherit those values from the parent profile.

Using a custom profile as the parent profile

When creating a custom profile, you can specify another custom profile, rather than the default profile, as the parent profile. The only restriction is that the custom profile that you specify as the parent must be of the same profile type as the child. Once you have created the child profile, its settings and default values are automatically those of the custom profile that you specified as the parent.

For example, if you create a profile called **my_http_profile2**, you can specify the custom profile **my_http_profile** as its parent. The result is that the default setting values of profile **my_http_profile2** are those of its parent profile **my_http_profile**.

If you subsequently modify the settings of the parent profile, the LTM system automatically propagates those changes to the child profile. For example, if you create the custom profile **my_http_profile** and use it as a parent profile to create the custom profile **my_http_profile2**, any changes you make later to profile **my_http_profile** are automatically propagated to profile **my_http_profile2**. Again, the exception to this is when you modify any of the settings in the child profile. For those modified settings, the child profile does not inherit the values of its parent.

Summarizing profiles

Profiles are a configuration tool that you can use to affect the behavior of certain types of network traffic. By default, the LTM system provides you with a set of profiles that you can use as is. These profiles contain various settings that define the behavior of FastL4, TCP, UDP, RSTP, HTTP, FTP, and SSL traffic. Profiles also give you a way to enable connection and session persistence, and to manage client application authentication. Once you have assigned a profile to a virtual server, the LTM system manages any traffic that corresponds to that profile type according to the settings defined in that profile.

There are two possible types of profiles: default profiles, which the LTM system supplies, and custom profiles, which you create. Default profiles are useful when the values contained in them are sufficient for your needs. Custom profiles are useful when you want your values to differ from those contained in the default profile. To ease your task of configuring and maintaining profiles, the LTM system ensures that a custom profile automatically inherits settings and values from a parent profile.

When you create profiles to manage a type of network traffic, you can use them in the following ways:

- You do not need to take any action to use the default profiles that are enabled by default. The LTM system uses them to automatically direct the corresponding traffic types according to the values specified in the those profiles.
- You can create a custom profile, using the default profile as the parent profile, modifying some or all of the values defined in that profile.
- You can create a custom profile to use as a parent profile for other custom profiles.

Creating and modifying profiles

As described in the previous section, profiles are a configuration tool to help you manage your application traffic. To make use of profiles, you can either use the default profiles that the LTM system provides, or you can create your own custom profiles. You can also modify existing profiles as needed.

More specifically, you can:

- Use a default profile as is.
- Modify a default profile.
- Create a custom profile.
- Modify a custom profile.

The following sections contain the procedures for creating and modifying profiles. To understand individual profile settings and their effect on different types of traffic, see either the remainder of this chapter, or one of the following chapters:

- Chapter 6, *Managing HTTP and FTP Traffic*
- Chapter 7, *Managing SSL Traffic*
- Chapter 8, *Authenticating Application Traffic*
- Chapter 9, *Enabling Session Persistence*

For background information on default and custom profiles, see *Introducing profiles*, on page 5-1.

Using a default profile as is

The LTM system provides a default profile that you can use as is for each type of traffic. A default profile includes default values for any of the properties and settings related to managing that type of traffic. To implement a default profile, you simply assign the profile to a virtual server, using the Configuration utility. You are not required to configure the setting values. For more information, see *Implementing a profile*, on page 5-10.

For information on creating or modifying a virtual server, see Chapter 2, *Configuring Virtual Servers*.

Modifying a default profile

Using the Configuration utility, you can modify the values of a default profile. We do not recommend this. Although modifying a default profile appears to be simpler and quicker than creating a custom profile, be aware that in so doing, you lose the original values. If you want to reset the profile back to its original state, you must do this manually by modifying the

settings of the default profile again to specify the original values. (To find the original default values, see the relevant profile chapter in this guide, or see the online help.)

Modifying and implementing a default profile is a two-step process:

- First, you must modify the settings of the default profile, using the Configuration utility. For more information, see *To modify a default profile*, following.
- Second, you must associate that profile with a virtual server. For information on associating a profile with a virtual server, see *Implementing a profile*, on page 5-10.

To modify a default profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The HTTP Profiles screen opens.
3. Select the default profile that you want to modify:
   - If you are modifying the **http** profile, click the name **http**.
     This displays the properties and settings of the default **http** profile.
   - If you are modifying a default profile other than the **http** profile, click the appropriate profile menu on the menu bar and choose a profile type. Then click a profile name.
     This displays the properties and settings of that default profile.
4. Modify the settings to suit your needs.
5. Click **Update**.

Creating a custom profile

If you do not want to use a default profile as is or change its settings, you can create a custom profile. Creating a custom profile and associating it with a virtual server allows you to implement your own specific set of traffic-management policies.

When you create a custom profile, the profile is a child profile and automatically inherits the setting values of a parent profile that you specify. However, you can change any of the values in the child profile to better suit your needs. For background information on custom profiles and inheritance of setting values, see *Custom and parent profiles*, on page 5-3.

If you do not specify a parent profile, the LTM system uses the default profile that matches the type of profile you are creating.

Implementing a custom profile is a two-step process:

- First, you must create the custom profile, using the Configuration utility. For more information, see *To create a custom profile*, following.
- Second, you must associate that profile with a virtual server. For information on associating a profile with a virtual server, see *Implementing a profile*, on page 5-10.

Important

Within the Configuration utility, each profile creation screen contains a check box to the right of each profile setting. When you check a box for a setting and then specify a value for that setting, the profile then retains that value, even if you change the corresponding value in the parent profile later. Thus, checking the box for a setting ensures that the parent profile never overwrites that value through inheritance.

To create a custom profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens and, by default, displays a list of any existing HTTP profiles.
3. Select the type of profile you want to create:
   - If you are creating an HTTP type of profile, proceed to Step 4.
   - If you are creating another type of profile, click a profile category on the menu bar and choose a profile type.
4. On the right side of the screen, click **Create**.
   This displays the screen to create a new profile.
5. In the **Name** box, type a unique name for your profile.
6. For the **Parent Profile** setting, select a profile from the list.
   You can select either the default profile or another custom profile.
7. Specify, modify, or retain values for all settings:
   - If you want to specify or modify a value, locate the setting, click the box in the Custom column on the right side of the screen, and then type or modify a value.
   - If you want to retain a value inherited from the parent profile, leave the setting as is. Do not check the box in the Custom column.
8. Click **Finished**.

Tip

*An alternative way to access the New Profile screen is to expand **Local Traffic** On the Main tab, click the Create button next to the **Profiles** menu item, and select a profile type.*

Modifying a custom profile

Once you have created a custom profile, you can use the Configuration utility to adjust the settings of your custom profile later if necessary. If you have already associated the profile with a virtual server, you do not need to perform that task again.

Important

Within the Configuration utility, each profile creation screen contains a check box to the right of each profile setting. When you check a box for a setting and then specify a value for that setting, the profile then retains that value, even if you change the corresponding value in the parent profile later. Thus, checking the box for a setting ensures that the parent profile never overwrites that value through inheritance.

To modify custom profile settings

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The HTTP Profiles screen opens.
3. Point to the menu for the type of profile you want to modify (Services, Persistence, Protocols, SSL, or Authentication) and choose a profile type.
   This displays a list of existing profiles of that type.
4. In the Name column, click the name of the profile you want to modify.
   This displays the settings and values for that profile.
5. Modify or retain values for all settings:
   - If you want to modify a value, locate the setting, click the box in the Custom column on the right side of the screen, and then modify the value.
   - If you want to retain a value inherited from the parent profile, leave the setting as is. Do not check the box in the Custom column.
   - If you want to reset a value back to the parent profile value, clear the check box in the Custom column on the right side of the screen.
6. Click the **Update** button.

Implementing a profile

Once you have created a profile for a specific type of traffic, you implement the profile by associating that profile with one or more virtual servers.

You associate a profile with a virtual server by configuring the virtual server to reference the profile. Whenever the virtual server receives that type of traffic, the LTM system applies the profile settings to that traffic, thereby controlling its behavior. Thus, profiles not only define capabilities per network traffic type, but also ensure that those capabilities are available for a virtual server.

To assign a profile to a virtual server

1. On the Main tab, expand **Local Traffic**.
2. Click **Virtual Servers**.
   This displays a list of existing virtual servers.
3. Click a virtual server name.
   This displays the properties and settings for that virtual server.
4. Locate the setting for the type of profile you want to assign and select the name of a default or custom profile.
5. At the bottom of the screen, click **Update**.

Note

You can also assign a profile to a virtual server at the time that you create the virtual server.

Because certain kinds of traffic use multiple protocols and services, users often create multiple profiles and associate them with a single virtual server.

For example, a client application might use the TCP, SSL, and HTTP protocols and services to send a request. This type of traffic would therefore require three profiles, based on the three profile types TCP, Client SSL, and HTTP.

Each virtual server lists the names of the profiles currently associated with that virtual server. You can add or remove profiles from the profile list, using the Configuration utility.

The LTM system has specific requirements regarding the combinations of profile types allowed for a given virtual server. Table 5.2 shows the specific combinations of profile types that you can configure on a virtual server.

Profile Type	Prerequisite Profiles	Incompatible Profiles
Protocol profiles		
Fast L4	None	All
Fast HTTP	None	All
TCP	None	UDP, Fast L4, Fast L7
UDP	None	TCP, Fast L4, Fast L7
Services profiles		
HTTP	TCP	FTP
FTP	TCP	HTTP, CLient SSL or Server SSL
SSL profiles		
Client SSL	TCP	FTP
Server SSL	TCP	FTP
Persistence profiles		
Cookie	HTTP	N/A
Destination Address Affinity	Any	None
Hash	Fast L4, TCP, UDP	N/A
MSRDP	TCP	N/A
SIP	TCP or UDP	FTP
Source Address Affinity	Any	None
SSL	TCP	FTP
Universal	None	N/A
Authentication profiles		
LDAP	TCP	N/A
RADIUS	TCP	N/A
TACACS+	TCP	N/A
SSL Client Certificate LDAP	TCP	N/A
OCSP	TCP	N/A

Table 5.2 *Profile combinations that the LTM system allows and disallows*

Profile Type	Prerequisite Profiles	Incompatible Profiles
Other profiles		
OneConnect	TCP	N/A
Statistics	TCP	N/A
Stream	TCP	Fast L4, UDP

Table 5.2 *Profile combinations that the LTM system allows and disallows*

In directing traffic, if a virtual server requires a specific type of profile that does not appear in its profile list, the LTM system uses the relevant default profile, automatically adding the profile to the profile list. For example, if a client application sends traffic over TCP, SSL, and HTTP, and you have assigned SSL and HTTP profiles only, the LTM system automatically adds the default profile **tcp** to its profile list.

At a minimum, a virtual server must reference a profile, and that profile must be associated with a UDP, FastL4, Fast HTTP, or TCP profile type. Thus, if you have not associated a profile with the virtual server, the LTM system adds a UDP, FastL4, Fast HTTP, or TCP default profile to the profile list.

The default profile that the LTM system chooses depends on the configuration of the virtual server's protocol setting. If the protocol setting is set to **UDP**, the LTM system adds the **udp** profile to its profile list. If the protocol setting is set to anything other than **UDP**, the LTM system adds the **FastL4** profile to its profile list.

Configuring protocol-type profiles

Some of the profiles that you can configure are known as Protocol profiles. The Protocol profiles types are:

- Fast L4
- Fast HTTP
- TCP
- UDP

For each Protocol profile type, the LTM system provides a pre-configured profile with default settings. In most cases, you can use these default profiles as is. If you want to change these settings, you can configure protocol profile settings when you create a profile, or after profile creation by modifying the profile's settings.

The remainder of this section lists the traffic-management settings contained in the Fast L4, Fast HTTP, TCP, and UDP profiles. For information on configuring other types of profiles, see the following:

- For information on the OneConnect, Statistics, and Stream profiles, see *Configuring other types of profiles*, on page 5-24.
- For information on the HTTP and FTP profiles, see Chapter 6, *Managing HTTP and FTP Traffic*.
- For information on the Client SSL and Server SSL profiles, see Chapter 7, *Managing SSL Traffic*.
- For information on the profiles used for remote authentication, see Chapter 8, *Authenticating Application Traffic*.
- For information on the profiles used for session persistence, see Chapter 9, *Enabling Session Persistence*.

The Fast L4 profile type

The purpose of a Fast L4 profile is to help you manage Layer 4 traffic more efficiently. When you assign a Fast L4 profile to a virtual server, the Packet Velocity ASIC® (PVA) hardware acceleration within the BIG-IP system can process some or all of the Layer 4 traffic passing through the system. By offloading Layer 4 processing to the PVA hardware acceleration, the BIG-IP system can increase performance and throughput for basic routing functions (Layer 4) and application switching (Layer 7).

You can use a Fast L4 profile with these types of virtual servers--Performance (Layer 4), Forwarding (Layer 2), and Forwarding (IP). Therefore, you can use a Fast L4 profile when you do *not* need the following traffic management features:

- HTTP optimizations
- TCP optimizations
- OneConnect™

- iRules™ for non-Layer 4 events
- Session persistence types other than source address affinity or destination address affinity persistence
- HTTP data compression
- Remote authentication
- HTTP pipelining

For your typical needs, most of the Fast L4 profile settings suffice. The specific settings that you might want to change are **Reset on Timeout** and **Idle Timeout**.

◆ Note

*Any changes you make to an existing Fast L4 profile take effect on a connection only after the **Idle Timeout** value has expired or the connection is closed.*

Table 5.3 lists and describes the settings of the Fast L4 profile type.

Setting	Description	Default Value
Name	This setting specifies a unique name for the profile.	No default value
Parent Profile	This setting specifies the profile that you want to use as the parent profile. Your new profile inherits all non-custom settings and values from the parent profile specified.	**fastL4**
Reset on Timeout	If this setting is enabled and a TCP connection exceeds the timeout value for idle connections, the LTM system sends a reset in addition to deleting the connection.	**Enable**
Reassemble IP Fragments	If this setting is enabled, the LTM system reassembles IP fragments.	**Disable**
Idle Timeout	This setting specifies the number of seconds that a connection is idle before the connection is eligible for deletion.	**300**
Max Segment Size Override	If set to a non-zero value, this setting overrides the maximum segment size of **1460**.	**0**
PVA Acceleration	This setting specifies the maximum acceleration mode that you *prefer* the system to use. Note that depending on the virtual server configuration, the system might or might not accelerate traffic in this mode. Possible values are **Full**, **Assisted**, or **None**. Additional information on this setting follows this table.	**Full**
IP ToS to Client	This setting specifies the Type of Service level that the LTM system assigns to UDP packets when sending them to clients.	**65535**
IP ToS to Server	This setting specifies the Type of Service level that the LTM system assigns to UDP packets when sending them to servers	**65535**

***Table 5.3** Settings of a Fast L4 profile*

Setting	Description	Default Value
Link QoS to Client	This setting specifies the Quality of Service level that the LTM system assigns to UDP packets when sending them to clients.	**65535**
Link QoS to Server	This setting specifies the Quality of Service level that the LTM system assigns to UDP packets when sending them to servers.	**65535**
TCP Timestamp Mode	Specifies the action that the LTM system should take on TCP timestamps. Possible values are: **Preserve**, **Strip**, and **Rewrite**.	**Preserve**
TCP Window Scale Mode	Specifies the action that the LTM system should take on TCP windows. Possible values are: **Preserve**, **Strip**, and **Rewrite**.	**Preserve**
Generate Internal Sequence Numbers	Enables the LTM system to generate its own sequence numbers for SYN packets, according to RFC 1948. When enabled, this setting allows timestamp recycling.	Disabled
Strip Sack OK	Enables the LTM system to block a TCP SackOK option from passing to the server on an initiating SYN.	Disabled
RTT from Client	Specifies that the LTM system should use TCP timestamp options to measure the round-trip time to the client.	Disabled
RTT from Server	Specifies that the LTM system should use TCP timestamp options to measure the round-trip time to the server.	Disabled

Table 5.3 *Settings of a Fast L4 profile*

Once you implement a Fast L4 profile, the BIG-IP system automatically selects the most efficient PVA hardware acceleration mode for Layer 4 traffic. Possible modes are **Full**, **Assisted**, and **None**.

The particular hardware acceleration mode that the BIG-IP system selects depends on these factors:

- **The Fast L4 profile settings**
  The mode that the BIG-IP selects is influenced by the way that you configure the settings of the Fast L4 profile.
- **The virtual server configuration**
  The mode that the BIG-IP system selects is influenced by the specific features that you assigned to the virtual server (such as pools, SNAT pools, and iRules).
- **The value of the PVA Acceleration setting**
  The **PVA Acceleration** setting in the Fast L4 profile defines the maximum amount of hardware acceleration that you want to allow, for Layer 4 traffic passing through the virtual server. Therefore, if you set the value to:
  - **Full** (the default value)--The system can set hardware acceleration to any of the three modes, depending on the virtual server configuration.
  - **Assisted**--The system can set hardware acceleration to either **Assisted** or **None** mode, depending on the virtual server configuration.

- **None**--The system does not perform hardware acceleration.

One reason that you might want to set the maximum hardware acceleration mode to less than **Full** is for viewing connections with the **bigpipe conn show** command. This command only shows Layer 4 connections when the hardware acceleration mode is set to **Assisted** or **None**. If the mode is set to **Full**, the **bigpipe conn show** command shows no Layer 4 connections.

Depending on the current mode to which hardware acceleration is automatically set, the BIG-IP system accelerates Layer 4 traffic as described in Table 5.4.

Hardware Acceleration Mode	Result
Full	The hardware acceleration processes all Layer 4 traffic. Layer 4 traffic is *not* managed through the use of BIG-IP software features. In this case, the BIG-IP system treats client-side and server-side packets as part of the same connection. An example of using hardware acceleration in **Full** mode is when you want to load balance Layer 4 traffic to two servers, using the Round Robin load balancing method, with no session persistence or iRules.
Assisted	The BIG-IP system load balances all SYN packets, while the hardware acceleration assists with the remaining packets, including the tearing down of connections. An example of using hardware acceleration in **Assisted** mode is when you want to load balance Layer 4 traffic using a dynamic load balancing method, or using a simple iRule that examines the IP addresses contained in the packets. ***Note:*** *When the BIG-IP system sets the hardware acceleration mode to* ***Assisted****, a Fast L4 profile is compatible with SNATs and SNAT pools, as well as with source address affinity persistence.*
None	The hardware acceleration does not process any Layer 4 traffic. The BIG-IP application manages all Layer 4 traffic. In this case, the BIG-IP system treats client-side and server-side packets as separate connections. An example of using hardware acceleration in **None** mode is when you want to load balance traffic using an HTTP profile, as well as an iRule that performs delayed binding and cookie session persistence.

Table 5.4 *Effect of PVA hardware acceleration mode on Layer 4 traffic*

The Fast HTTP profile type

The Fast HTTP profile is a configuration tool designed to speed up certain types of HTTP connections. This profile combines selected features from the TCP, HTTP, and OneConnect profiles into a single profile that is optimized for the best possible network performance. When you associate this profile with a virtual server, the virtual server processes traffic packet-by-packet, and at a significantly higher speed.

You might consider using a Fast HTTP profile when:

- You do not need features such as session persistence, remote server authentication, SSL traffic management, and TCP optimizations, nor HTTP features such as data compression, pipelining, and RAM Cache.
- You do not need to maintain source IP addresses.
- You want to reduce the number of connections that are opened to the destination servers.
- The destination servers support connection persistence, that is, HTTP/1.1, or HTTP/1.0 with **Keep-Alive** headers. Note that IIS servers support connection persistence by default.
- You need basic iRule support only (such as limited Layer 4 support and limited HTTP header operations). For example, you can use the iRule events **CLIENT_ACCEPTED**, **SERVER_CONNECTED**, and **HTTP_REQUEST**.

A significant benefit of using a Fast HTTP profile is the way in which the profile supports connection persistence. Using a Fast HTTP profile ensures that for client requests, the BIG-IP system can transform or add an HTTP **Connection** header to keep connections open. Using the profile also ensures that the BIG-IP system pools any open server-side connections. This support for connection persistence can greatly reduce the load on destination servers by removing much of the overhead caused by the opening and closing of connections. For more information on HTTP header transformation, see Chapter 6, *Managing HTTP and FTP Traffic*. For more information on the pooling of server-side connections, see *The OneConnect profile type*, on page 5-24.

◆ Note

The Fast HTTP profile is incompatible with all other profile types. Also, you cannot use this profile type in conjunction with VLAN groups, or with the IPv6 address format.

Table 5.5 lists and describes the settings of a Fast HTTP profile type.

Setting	Description	Default Value
Name	This setting specifies a unique name for the profile.	No default value
Parent Profile	This setting specifies the profile that you want to use as the parent profile. Your new profile inherits all non-custom settings and values from the parent profile specified.	fasthttp
Reset on Timeout	Specifies, when checked (enabled), that the system sends a TCP RESET packet when a connection times out, and deletes the connection.	Enabled (checked)
Idle timeout	This setting specifies the number of seconds that a connection is idle before the connection flow is eligible for deletion because it has no traffic. Possible values are: **Specify**, **Immediate**, and **Indefinite**. For more information, see the online help.	**300**
Maximum Segment Size Override	Specifies a maximum segment size (MSS) override for server-side connections. The default setting is **0**, which corresponds to an MSS of **1460**. You can specify any integer between **536** and **1460**.	**0**
Client Close Timeout	Specifies the number of seconds after which the system closes a client connection, when the system either receives a client FIN packet or sends a FIN packet to the client. This setting overrides the **Idle Timeout** setting. Possible values are: **Specify**, **Immediate**, and **Indefinite**. For more information, see the online help.	**5**
Server Close Timeout	Specifies the number of seconds after which the system closes a client connection, when the system either receives a server FIN packet or sends a FIN packet to the server. This setting overrides the **Idle Timeout** setting. Possible values are: **Specify**, **Immediate**, and **Indefinite**. For more information, see the online help.	**5**
Unclean Shutdown	Specifies how the system handles closing connections. Possible values are: **Disabled**, **Enabled**, and **Fast**. For more information, see the online help.	**Disabled**
Force HTTP 1.0 Response	Specifies, when checked (enabled), that the server sends responses to clients in the HTTP/1.0 format. This effectively disables client chunking and pipelining.	Disabled (unchecked)
Maximum Pool Size	Specifies the maximum number of connections a load balancing pool can accept. A setting of **0** specifies that a pool can accept an unlimited number of connections.	**2048**
Minimum Pool Size	Specifies the minimum number of connections that a load balancing pool can accept. A setting of **0** specifies that there is no minimum.	**0**
Ramp-Up Increment	Specifies the increment in which the system makes additional connections available, when all available connections are in use.	**4**
Maximum Reuse	Specifies the maximum number of times that the system can re-use a current connection.	**0**

Table 5.5 *Settings of a Fast HTTP profile*

Setting	Description	Default Value
Idle Timeout Override	Specifies the number of seconds after which a server-side connection in a pool is eligible for deletion, when the connection has no traffic. This setting overrides the **Idle Timeout** setting. Possible values are: **Specify**, **Disabled**, and **Indefinite**. For more information, see the online help.	**Disabled**
Replenish	Specifies whether the LTM system should maintain a steady-state maximum number of back-end connections. If you disable this setting, the system does not keep a steady-state maximum of connections to the back end, unless the number of connections to the pool drops below the value specified in the **Minimum Pool Size** setting.	Enabled (checked)
Parse Requests	Specifies, when checked (enabled), that the system parses the HTTP data in the connection stream. Note that if you are using a Fast HTTP profile for non-HTTP traffic, you should disable this setting to shield against dynamic denial-of-service (DDOS) attacks.	Enabled (checked)
Maximum Header Size	Specifies the maximum amount of HTTP header data that the system buffers before making a load balancing decision.	**32768**
Maximum Requests	Specifies the maximum number of requests that the system allows for a single client-side connection. When the specified limit is reached, the final response contains a **Connection: close** header is followed by the closing of the connection. The default setting of **0** means that the system allows an infinite number of requests per client-side connection.	**0**
Insert XForwarded For	Specifies whether the system inserts the **XForwarded For:** header in an HTTP request with the client IP address, to use with connection pooling. Possible settings are **Enabled** and **Disabled**. For more information, see the online help.	**Disabled**
Header Insert	Specifies a string that the system inserts as a header in an HTTP request. If the header exists already, the system does not replace it.	No default value

Table 5.5 *Settings of a Fast HTTP profile*

When writing iRules™, you can specify a number of events and commands that the Fast HTTP profile supports. The iRule events that the Fast HTTP profile supports are:

- **CLIENT_ACCEPTED**
- **SERVER_CONNECTED**
- **HTTP_REQUEST**

The iRule commands that the Fast HTTP profile supports are:

- **HTTP::method**
- **HTTP::uri**

- **HTTP::version**
- **HTTP::header exists**
- **HTTP::header value**
- **HTTP::header insert**

For more information about these iRule events and commands, see Chapter 13, *Writing iRules*.

The TCP profile type

The TCP profile is a configuration tool for managing TCP network traffic. Many of the TCP profile settings are standard SYSCTL types of settings, while others are unique to the LTM system.

A TCP profile is important because it is required for implementing certain types of other profiles. For example, by implementing the TCP and HTTP profiles, along with a persistence profile and a remote authentication profile, you can take advantage of these traffic management features:

- Content spooling to reduce server load
- OneConnect, which pools server-side connections
- Layer 7 session persistence, such as hash or cookie persistence
- iRules for managing HTTP traffic
- HTTP RamCache
- HTTP data compression
- HTTP pipelining
- Application authentication using a remote server
- Rewriting of HTTP redirections

Table 5.6 lists and describes the settings of a TCP profile type.

Setting	Description	Default Value
Name	This setting specifies a unique name for the profile.	No default value
Parent Profile	This setting specifies the profile that you want to use as the parent profile. Your new profile inherits all non-custom settings and values from the parent profile specified.	**tcp**
Reset on Timeout	If this setting is enabled and a TCP connection exceeds the timeout value for idle connections, the LTM system sends a reset in addition to deleting the connection.	Enabled
Time Wait Cycle	This setting recycles the connection when a SYN packet is received in a TIME-WAIT state.	Enabled
Delayed ACKs	If this setting is enabled, the LTM system allows coalescing of multiple acknowledgement (ACK) responses.	Enabled

Table 5.6 *Settings of a TCP profile*

Setting	Description	Default Value
Proxy Maximum Segment	Advertises the same maximum segment to the server as was negotiated with the client.	Enabled
Proxy Options	Advertises an option (such as timestamps) to the server only if it was negotiated with the client.	Disabled
Proxy Buffer Low	Specifies the proxy buffer level at which the receive window was opened.	**4096**
Proxy Buffer High	Specifies the proxy buffer level at which the receive window was closed.	**16384**
Idle Timeout	This setting specifies the number of seconds that a connection is idle before the connection is eligible for deletion.	**300**
Time Wait	This setting specifies the number of milliseconds that a connection is in a TIME-WAIT state before entering the CLOSED state.	**2000**
FIN Wait	This setting specifies the number of seconds that a connection is in the FIN-WAIT or CLOSING state before quitting. A value of **0** represents a term of forever (or until the metrics of the FIN state).	**5**
Close Wait	This setting specifies the number of seconds that a connection remains in a LAST-ACK state before quitting. A value of **0** represents a term of forever (or until the metrics of the FIN state).	**5**
Send Buffer	This setting causes the LTM system to send the buffer size, in bytes.	**8192**
Receive Window	This setting causes the LTM system to receive the window size, in bytes.	**32768**
Keep Alive Interval	This setting causes the LTM system to keep alive the probe interval, in milliseconds.	**1800**
Maximum SYN Retransmissions	This setting specifies the maximum number of retransmissions of SYN segments that the LTM system allows.	**4**
Maximum Segment Retransmissions	This setting specifies the maximum number of retransmissions of data segments that the LTM system allows.	**8**
IP ToS	This setting specifies the Type of Service level that the LTM system assigns to TCP packets when sending them to clients.	**0**
Link QoS	This setting specifies the Quality of Service level that the LTM system assigns to TCP packets when sending them to clients.	**0**
Selected ACKs	This setting specifies, when checked (enabled), that the system processes data using selective ACKs whenever possible, to improve system performance.	Enabled (checked)

Table 5.6 *Settings of a TCP profile*

Setting	Description	Default Value
Extended Congestion Notification	This setting specifies, when checked (enabled), that the system uses the TCP flags **CWR** and **ECE** to notify its peer of congestion and congestion counter-measures.	Disabled (unchecked)
Extensions for High Performance (RFC 1323)	This setting specifies, when checked (enabled), that the system uses the timestamp and window scaling extensions for TCP (as specified in RFC 1323) to enhance high-speed network performance.	Enabled (checked)
Limited Transmit Recovery	This setting specifies, when checked (enabled), that the system uses limited transmit recovery revisions for fast retransmits (as specified in RFC 3042) to reduce the recovery time for connections on a lossy network.	Enabled (checked)
Slow Start	This setting specifies, when checked (enabled), that the system uses larger initial window sizes (as specified in RFC 3390) to help reduce round trip times.	Enabled (checked)
Deferred Accept	This setting specifies, when checked (enabled), that the system defers allocation of the connection chain context until the system has received the payload from the client. Enabling this setting is useful in dealing with 3-way handshake denial-of-service attacks.	Disabled (unchecked)
Bandwidth Delay	This setting specifies, when checked (enabled), that the system attempts to calculate the optimal bandwidth to use to the client, based on throughput and round-trip time, without exceeding the available bandwidth.	Enabled (checked)
Nagle's Algorithm	Specifies, when checked (enabled), that the system applies Nagle's algorithm to reduce the number of short segments on the network. The default setting is disabled. Note that enabling this setting for interactive protocols such as telnet may cause degradation on high-latency networks.	Enabled (checked)

Table 5.6 *Settings of a TCP profile*

For most of the TCP profile settings, the default values usually meet your needs. However, if the link that clients are using to access the virtual server is slow, or if server response time exceeds the request time of clients, you can increase the content spooling settings of the profile:

- **Proxy Buffer Low**
- **Proxy Buffer High**
- **Send Buffer**
- **Receive Window**

Increasing the byte values of the these settings increases the amount of data that the BIG-IP system can buffer while waiting for a specific connection to accept that data.

◆ Note

*If you are using a TCP profile in a test environment, you can improve performance by disabling the **Slow Start**, **Bandwidth Delay**, and **Nagle's Algorithm** settings.*

The UDP profile type

The UDP profile is a configuration tool for managing UDP network traffic. Table 5.7 lists and describes the settings of a UDP profile type.

Setting	Description	Default Value
Name	This setting specifies a unique name for the profile.	No default value
Parent Profile	This setting specifies the profile that you want to use as the parent profile. Your new profile inherits all non-custom settings and values from the parent profile specified.	udp
Idle timeout	This setting specifies the number of seconds that a connection is idle before the connection flow is eligible for deletion.	**60**
IP ToS	This setting specifies the Type of Service level that the LTM system assigns to UDP packets when sending them to clients.	0
Link QoS	This setting specifies the Quality of Service level that the LTM system assigns to UDP packets when sending them to clients.	0
Datagram LB	This setting specifies, when checked (enabled), that the system load balances UDP traffic packet-by-packet.	Disabled (unchecked)

***Table 5.7** Settings of a UDP profile*

Configuring other types of profiles

Some of the other profile types that you can configure are:

- OneConnect
- Statistics
- Stream

For each Protocol profile type, the LTM system provides a pre-configured profile with default settings. In most cases, you can use these default profiles as is. If you want to change these settings, you can configure protocol profile settings when you create a profile, or after profile creation by modifying the profile's settings.

The following sections list the traffic-management settings contained in OneConnect, Statistics, and Stream profiles. For information on configuring other types of profiles, see the following:

- For information on the Fast L4, Fast HTTP, TCP, and UDP profiles, see *Configuring protocol-type profiles*, on page 5-13.
- For information on the HTTP and FTP profiles, see Chapter 6, *Managing HTTP and FTP Traffic*.
- For information on the SSL profiles, see Chapter 7, *Managing SSL Traffic*.
- For information on the profiles for session persistence, see Chapter 9, *Enabling Session Persistence*.
- For information on the profiles for remote authentication, see Chapter 8, *Authenticating Application Traffic*.

The OneConnect profile type

The OneConnect™ profile is a configuration tool for enabling connection pooling on an LTM system. ***Connection pooling*** optimizes the way that the LTM system handles connections. When connection pooling is enabled on an LTM system, client requests can utilize existing, server-side connections, thus reducing the number of server-side connections that a server must open to service those requests.

The LTM system can pool connections from multiple virtual servers if those virtual servers reference the same OneConnect profile and the same pool. Table 5.8 lists and describes the settings of a OneConnect profile type.

◆ Tip

You can also enable a related feature known as the OneConnect Transformations feature. You enable this feature from within an HTTP profile. The ***OneConnect Transformations*** *HTTP profile setting applies to HTTP/1.0 connections, and when enabled, causes the system to transform the value of the* ***Connection*** *header in an HTTP request to* ***Keep-Alive****, to keep a connection open. This feature, together with a OneConnect profile, optimizes connection persistence.*

Setting	Description	Default Value
Name	This setting specifies a unique name for the profile.	No default value
Parent Profile	This setting specifies the profile that you want to use as the parent profile. Your new profile inherits all non-custom settings and values from the parent profile specified.	**oneconnect**
Source Mask	The LTM system applies the value of this setting to the source address to determine its eligibility for reuse. A mask of **0** causes the LTM system to share reused connections across all clients. A host mask (that is, all **1** values in binary), causes the LTM system to share only those reused connections originating from the same client IP address.	**0.0.0.0**
Max Size	The setting defines the maximum number of connections that the LTM system holds in the connection reuse pool. If the pool is already full, then a server-side connection closes after the response is completed.	**10000**
Max Age	This setting defines the maximum number of seconds allowed for a connection in the connection reuse pool. For any connection with an age higher than this value, the LTM system removes that connection from the resue pool.	**86400**
Max Reuse	This setting specifies the maximum number of times that a server-side connection can be reused.	**1000**
Idle Timeout Override	This setting specifies the number of seconds that a connection is idle before the connection flow is eligible for deletion. You can use this setting to increase the timeout value for connections once they are pooled for re-use. Possible values are **Disabled**, **Indefinite**, or a numeric value that you specify.	**Disabled**

Table 5.8 *Settings of a OneConnect profile*

The Statistics profile type

The Statistics profile provides user-defined statistical counters. Each profile contains 32 settings (**Field1** through **Field32**), which define named counters. Using a Tcl-based iRule command, you can use the names to manipulate the counters while processing traffic.

For example, you can assign the counters **tot_users**, **cur_users**, and **max_users** to the Statistics profile settings **Field1**, **Field2**, and **Field3** respectively, and then write an iRule as shown in Figure 5.1:

```
rule track_users {
when CLIENT_ACCEPTED {
  STATS::incr tot_users
  STATS::setmax users max_users [STATS:incr users cur_users]
  }
  when CLIENT_CLOSED {
  STATS::incr users tot_users -1
  }
  }
  virtual users_tcp {
  profile tcp users
  rule users
  …
  }
```

Figure 5.1 *Example of Statistics profile counters used in an iRule*

In this example, the counter **tot_users** counts the total number of connections, the counter **cur_users** counts the current number of connections, and the counter **max_users** retains the largest value of the counter **cur_users**.

For information on iRules **STATS** commands, see Chapter 13, *Writing iRules*.

The Stream profile type

You can use the Stream profile to search and replace strings within a data stream, such as a TCP connection. Table 5.9 lists and describes the settings of a Stream profile type.

Setting	Description	Default Value
Name	This setting specifies a unique name for the profile.	No default value
Parent Profile	This setting specifies the profile that you want to use as the parent profile. Your new profile inherits all non-custom settings and values from the parent profile specified.	**stream**
Source	Specifies the source string for which to search.	No default value
Target	Specifies the target string to replace.	No default value

Table 5.9 *Settings of a Stream profile*

Managing profiles

Using the Configuration utility, you can not only create and implement profiles, but also:

- View the settings of an existing profile
- Delete a profile
- View or reset profile statistics

Viewing profiles

You can view profile settings and values, using the Configuration utility.

To view profile settings

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. Point to the menu for the type of profile you want to view (Services, Persistence, Protocol, SSL, or Authentication) and choose a profile type.
   This displays a list of existing profiles of that type.
4. In the Name column, click the name of the profile you want to view.
   This displays the settings and values for that profile.

Tip

*When listing existing profiles, you can use the **Search** box that appears directly above the profile list. With the **Search** box, you can specify a string to filter the list, thereby showing only those objects that match the string. The default setting is an asterisk (*), which means show all objects.*

Deleting profiles

You can delete an existing profile, using the Configuration utility, as long as the profile is not referenced by a virtual server.

To delete a profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.

3. Point to the menu for the type of profile you want to view (Services, Persistence, Protocol, SSL, or Authentication) and choose a profile type.
   This displays a list of existing profiles of that type.
4. In the Select column, check one or more boxes next to the names of the profiles you want to delete.
5. Click the **Delete** button.
   This displays the **Delete Confirmation** screen.
6. Verify that all check boxes in the list are checked, and click the **Delete** button to permanently delete those profiles.

Using profiles with iRules

In some cases, the best way to manage a particular type of connection is to create an iRule. A good example is when you want to insert a header into an HTTP request and then direct the request based on the information in that header.

An iRule is a user-written script that manages a particular traffic connection when the connection meets certain criteria. For example, you can write an iRule that states that if a header in an HTTP request contains a certain string, the LTM system should send that request to the pool **http_pool**. An iRule is triggered when an event occurs that is specified within that iRule. iRule events are categorized into specific types, such as TCP, SSL, and HTTP.

When an iRule event occurs, the LTM system cannot actually trigger the iRule unless the virtual server has a profile in its profile list that corresponds to that event type. For example, if an iRule specifies an HTTP event, the virtual server must reference a profile that is based on the HTTP profile type.

The following list shows the possible types of iRule events and their profile requirements.

- **IP events**
  No profile requirement
- **UDP events**
  Requires a UDP- or FastL4-based profile
- **TCP events**
  Requires a TCP- or FastL4-based profile
- **FTP events**
  Requires an FTP-based profile
- **HTTP events**
  Requires an HTTP- and a TCP-based profile
- **SSL events**
  Requires either a Client SSL- or Server SSL-based profile, depending on the iRule context
- **AUTH events**
  Requires an authentication profile

For more information on iRule events, see Chapter 13, *Writing iRules*.

6

Managing HTTP and FTP Traffic

- Introducing HTTP and FTP traffic management
- Configuring HTTP profile settings
- Configuring HTTP compression
- Configuring the RAM Cache
- Configuring FTP profile properties and settings
- Managing HTTP and FTP profiles

Introducing HTTP and FTP traffic management

The BIG-IP® local traffic management (LTM) system offers several features that you can use to intelligently control your HTTP, HTTPS, and FTP traffic. Examples of these features are the insertion of headers into HTTP requests and the compression of HTTP server responses.

These features are available through configuration of an HTTP or an FTP profile. A ***profile*** is a group of settings, with values, that corresponds to a specific type of traffic, such as HTTP traffic. A profile defines the way that you want the LTM system to manage that traffic type.

In addition to HTTP and FTP profiles, the LTM system includes other features to help you manage your application traffic, such as health monitors for checking the health of HTTP and FTP services, and iRules™ for querying or manipulating HTTP header or content data.

◆ Note

In addition to the HTTP profile, there is another type of HTTP profile called a FastHTTP profile. The use of a FastHTTP profile is incompatible with the use of an HTTP profile. For more information on the FastHTTP profile, see Chapter 5, ***Understanding Profiles****.*

Table 6.1 summarizes the capabilities within the LTM system for managing HTTP and FTP traffic and shows the LTM object that you configure to implement each feature.

.

Feature	Description	Configuration Object
HTTP compression	You can create an HTTP profile that causes the LTM system to compress server responses using well-defined algorithms such as **gzip** and **deflate**.	HTTP profile
Monitoring of HTTP, HTTPS, and FTP ports on pool members	You can associate an HTTP, HTTPS, or FTP monitor with the members of a pool to ensure that pool members are ready and able to receive traffic on specific ports.	Load balancing pool HTTP, HTTPS, and FTP health monitors
Header insertion and deletion	You can insert headers into HTTP requests, or delete content from HTTP headers.	HTTP profile and iRule
Session persistence	You can ensure that HTTP sessions persist to the same pool member across connections.	Persistence profile and iRule
Header and content inspection and modification	By writing an iRule, users can inspect or modify the content of an HTTP request or response.	iRule
Redirection of HTTP requests	By configuring an HTTP profile or writing an iRule, you can instruct the LTM system to redirect HTTP traffic based on URI, status code, or content.	HTTP Profile and iRule
Pooling connections	You can configure the LTM system so that multiple client connections can reuse the same idle server-side connection. You can also configure the LTM system to rewrite the HTTP connection headers to ensure that a connection stays open.	OneConnect profile
Chunking of requests and responses	You can configure the LTM system to unchunk or rechunk HTTP responses.	HTTP profile
Pipelining	The LM system supports HTTP pipelining.	HTTP profile
IPV4-to-IPV6 compatibility	Ensures compatibility between IP version 4 and IP version 6 clients and servers when using the FTP protocol.	FTP profile

Table 6.1 *Summary of the LTM system features related to HTTP and FTP traffic control*

Configuring HTTP profile settings

You can configure HTTP profile settings to ensure that HTTP traffic management suits your specific needs. These configuration settings are organized into several categories on the New HTTP Profile screen in the Configuration utility: General Properties, Settings, Compression, and RAM Cache. You can configure these settings when you create a profile, or after profile creation by modifying the profile's settings. For specific procedures on configuring a profile, see Chapter 5, *Understanding Profiles*.

For the profile settings that appear in the Properties and Settings areas of the HTTP Profile screen, you can specify values where none exist, or modify any default values to suit your needs. For information about other HTTP profile settings, see *Configuring HTTP compression*, on page 6-11, and *Configuring the RAM Cache*, on page 6-21.

Table 6.2 shows these profile settings. For those settings that have default values, you can retain those default settings or modify them. Following this table are descriptions of the settings and the procedure for changing them.

Setting	Description	Default Value
Name	Specifies the user-supplied name of the profile. You must specify a name for your profile.	No default value
Parent Profile	Specifies the profile from which your custom profile is derived.	**http**
Basic Auth Realm	Specifies an authentication realm for client authentication.	No default value
Fallback Host	Specifies the fallback host to send as an HTTP 302 response when all nodes are down.	No default value
Header Insert	Specifies the header string that you want to insert into an HTTP request.	No default value
Header Erase	Specifies the header string that you want to erase from an HTTP request.	No default value
Response Chunking	Specifies how to handle chunking for HTTP responses. Possible values are **Unchunk**, **Rechunk**, **Selective**, and **Preserve**.	**Preserve**
OneConnect Transformations	Performs HTTP header transformations for the purpose of keeping connections open. This feature requires configuration of a One Connect™ profile.	**Disabled**
Redirect Rewrite	Allows you to modify HTTP redirections. Possible values are **Matching**, **All**, **Nodes**, or **None**.	**None**
Maximum Header Size	Specifies the maximum size that the LTM system allows for HTTP headers. The default value is represented in bytes.	**32**

Table 6.2 *Configuration settings in an HTTP profile*

Setting	Description	Default Value
Pipelining	Enables or disables HTTP pipelining.	**Disabled**
Insert **XForwarded-For**	Specifies an **XForwarded-For** header that the LTM system can insert into an HTTP request, to use with connection pooling. This feature adds the IP address of the client as the value of the **XForwarded-For** header.	No default value
LWS Maximum Columns	Specifies the maximum width allowed for an HTTP header that is inserted into an HTTP request.	**80**
LWS Separator	Specifies the separator that the LTM system should use between HTTP headers when a header exceeds the maximum width allowed.	**\r\n**
Maximum Requests	Specifies the maximum number of HTTP requests that the system allows for a single **Keep-Alive** connection.	**0**

Table 6.2 *Configuration settings in an HTTP profile*

Before configuring an HTTP profile, it is helpful to have a description of certain settings that you might want to change.

Specifying a profile name

To create an HTTP profile, you must specify a unique name for the profile. The **Name** setting is one of only two settings for which you must actively specify a value when creating an HTTP profile; all other settings have default values.

To specify a profile name, simply locate the **Name** setting and type a unique name for the profile.

Specifying a parent profile

Every profile that you create is derived from a parent profile. You can use the default **http** profile as the parent profile, or you can use another HTTP profile that you have already created.

To specify a parent profile, locate the **Parent Profile** setting and select a profile name.

Specifying a realm for basic authentication

The value of the **Basic Auth Realm** setting is a string that you provide. The LTM system sends this string to a client as part of client authentication.

To configure this setting, locate the **Basic Auth Realm** setting and type a value.

Specifying a fallback host

Another feature that you can configure within an HTTP profile is HTTP redirection. ***HTTP redirection*** allows you to redirect HTTP traffic to another protocol identifier, host name, port number, or URI path. For example, if all members of the targeted pool are unavailable (that is, the members are disabled, marked as **down**, or have exceeded their connection limit), the LTM system can redirect the HTTP request to the fallback host, with the HTTP reply Status Code **302 Found**.

When configuring the LTM system to redirect HTTP traffic to a fallback host, you can specify an IP address or a fully-qualified domain name (FQDN). The value that you specify becomes the value of the **Location** header that the servers sends in the response. For example, you can specify a redirection as **http://redirector.siterequest.com**.

Inserting headers into HTTP requests

An optional setting in an HTTP profile is HTTP header insertion. 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 format of the header insertion that you specify is generally a quoted string. Alternatively, however, you can insert a Tools Command Language (TCL) expression into a header that dynamically resolves to the desired value. When you assign the configured HTTP profile to a virtual server, the LTM system then inserts the header specified in the profile into any HTTP request that the LTM system sends to a pool or pool member.

◆ Note

In addition to inserting a string such as a client IP address into an HTTP request, you can configure the LTM system to insert SSL-related headers into HTTP requests. Examples are: client certificates, cipher specifications, and client session IDs. To insert these types of headers, you must create an iRule. For more information on using iRule commands to perform header insertion, Chapter 13, ***Writing iRules****.*

To insert a header into an HTTP request, locate the **Header Insert** setting and type a value.

Erasing content from HTTP headers

Another optional setting is the **Header Erase** setting. Using this setting, you can configure a profile to erase the contents of a header from an HTTP request that is being sent from a client to a server. With this feature, you can erase header content 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.

When you use this setting, the LTM system erases the contents of the specified header and replaces that content with blank spaces. The header itself is retained.

◆ Note

This feature does not apply to HTTP responses being sent from a server to a client.

The client header with the contents to be erased must be specified as a quoted string. To erase a header from an HTTP request, locate the **Header Erase** setting and type a value.

Configuring chunking

Sometimes, you might want to inspect and/or modify HTTP application data, such as when you are using an iRule that inspects the content of an HTTP response. Such inspections or modifications require that the response be ***unchunked***, that is, not in chunked encoding. Using the **Response Chunking** settings, the LTM system can unchunk a chunked response before performing an action on that response.

Possible values for this setting are **Unchunk**, **Rechunk**, **Selective**, and **Preserve**. The default value is **Preserve**.

Table 6.3 describes each of these values and the action that the LTM system takes, depending on whether an original response is chunked or unchunked.

Setting	Original response is chunked	Original response is unchunked
Unchunk	The LTM system unchunks the response and processes the HTTP content, and passes the response on as unchunked. The connection closes when all data is sent to the client as indicated by the **Connection: Close** header.	The LTM system processes the HTTP content and passes the response on untouched.

Table 6.3 *Chunking behavior of the LTM system*

Setting	Original response is chunked	Original response is unchunked
Rechunk	The LTM system unchunks the response, processes the HTTP content, re-adds the chunk trailer headers, and then passes the response on as chunked. Any chunk extensions are lost.	The LTM system adds transfer encoding and chunking headers on egress.
Selective	Same as **Rechunk**.	The LTM system processes the HTTP content and then passes the response on untouched.
Preserve	The LTM system leaves the response chunked, processes the HTTP content, and passes the response on untouched. Note that if HTTP compression is enabled, the LTM system does not compress the response.	The LTM system processes the HTTP content and then passes the response on untouched.

Table 6.3 *Chunking behavior of the LTM system*

Enabling or disabling OneConnect transformations

This setting enables or disables part of the OneConnect™ feature. When enabled, this setting performs HTTP **Connection** header transformations on HTTP/1.0 requests, for the purpose of implementing **Keep-Alive** support for connection persistence. Thus, when a client sends an HTTP/1.0 request with a **Connection**: **Close** header, this feature forces the connection to remain open by transforming the header to **Connection: Keep-Alive**.

The default value for this setting is **Disabled**.

Important

To enable this setting, you must also enable the connection pooling component of the OneConnect™ feature. You enable connection pooling by configuring a OneConnect profile. For more information on connection pooling and configuring a OneConnect profile, see Chapter 5, ***Understanding Profiles****.*

For general information on the OneConnect™ feature, see Chapter 1, *Introduction to Local Traffic Management*.

To enable OneConnect transformations, locate the **OneConnect Transformations** setting and check the box.

Rewriting an 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.

Note that the rewriting of any redirection takes place only in the HTTP **Location** header of the redirection response, and not in any content of the redirection.

To enable the LTM system to rewrite HTTP redirections, you simply specify, through the Configuration utility, the way that you want the system to handle URIs during the rewrite. Once enabled, this feature rewrites the protocol name port number

Possible values for this setting are **Matching**, **All**, **Nodes,** or **None**.

Selecting URIs to rewrite

When configuring the LTM system to rewrite HTTP redirections, you specify whether the system should rewrite only those URIs matching the URI originally requested by the client (minus an optional trailing slash), or whether the system should rewrite all URIs. In the latter case, the system always rewrites redirected-to URIs, and rewrites those URIs as if they matched the originally-requested URIs.

If the URI contains a node IP address instead of a host name, you can configure the LTM system to change that IP address to the virtual server address.

Table 6.4 shows examples of how redirections of client requests are transformed when the LTM system is listening on port **443**, and the **Rewrite Redirections** setting is enabled.

Original Redirection	Rewrite of Redirection
http://www.myweb.com/myapp/	**https://www.myweb.com/myapp/**
http://www.myweb.com:8080/myapp/	**https://www.myweb.com/myapp/**

Table 6.4 *Examples of rewriting HTTP redirections with the system listening on port* ***443***

Table 6.5 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
http://www.myweb.com/myapp/	**https://www.myweb.com:4443/myapp/**
http://www.myweb.com:8080/myapp/	**https://www.myweb.com:4443/myapp/**

Table 6.5 *Examples of rewriting HTTP redirections with the system listening on port* ***4443***

Rewriting the protocol name

When configured to rewrite HTTP redirections, the LTM system 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 LTM system to rewrite the redirected URI to go to **https://www.sample.com/bar/** instead. (Note the addition of the trailing slash.)

Specifying the maximum header size

With this setting, you can specify the maximum size that the LTM system allows for HTTP headers. The default value is **32** and is represented in bytes.

Enabling support for pipelining

Normally, a client cannot make a request until the previous request has received a response. HTTP/1.1 pipelining allows clients to make requests even when prior requests have not received a response. For this to succeed, however, destination servers must include support for pipelining. This feature enables that support on the LTM system.

To enable pipelining, locate the **Pipelining** setting and check the box. By default, this feature is set to **Disabled**.

Inserting an XForwarded For header

When using connection pooling, which allows clients to make use of existing server-side connections, you can insert the **XForwarded For** header and specify a client IP address. When you configure the LTM system to insert this header, the target server can identify the request as coming from a client other than the client that initiated the connection.

Configuring the maximum columns for linear white space

This setting specifies the maximum number of columns allowed for a header that is inserted into an HTTP request.

To configure the **LWS Maximum Columns** setting, specify a maximum value.

Configuring a linear white space separator

This setting specifies the separator that the LTM system should use between HTTP headers when a header exceeds the maximum width specified by the **LWS Maximum Columns** setting.

To configure the **LWS Separator** setting, specify a value for the separator.

Specifying a maximum number of requests

The Maximum Requests setting specifies the maximum number of requests that the system allows for a single Keep-Alive connection. When the specified limit is reached, the final response contains a **Connection: close** header is followed by the closing of the connection. The default setting is **0**, which in this case means that the system allows an infinite number of requests per **Keep-Alive** connection.

Configuring HTTP compression

In a typical client-server scenario, browsers and servers can be configured to compress and uncompress HTTP content. HTTP compression reduces the amount of data to be transmitted, thereby significantly reducing bandwidth usage. The following two sections describe the benefits of configuring the LTM system to perform HTTP compression tasks.

Compression in a typical client-server scenario

When HTTP compression is enabled in a client-server environment, a browser inserts an **Accept-Encoding** header into a client request, specifying the compression methods that the browser understands. Consequently, the server reads the header and compresses the response body with one of those compression methods. The server then inserts a **Content-Encoding** header in the response, to communicate to the browser the compression method that the server used. Upon receiving the compressed response, the browser reads the **Content-Encoding** header and uncompresses the data accordingly.

Enabling HTTP compression without an LTM system typically requires installation and configuration of compression software on the destination server.

Compression using the LTM system

An optional feature is the LTM system's ability to off-load HTTP compression tasks from the target server. All of the tasks needed to configure HTTP compression on the LTM system, as well as the compression software itself, are centralized on the LTM system.

The primary way to enable the HTTP compression option is by setting the **Compression** setting of an HTTP profile to **Enabled**. This causes the LTM system to compress HTTP content for any responses matching the values that you specify in the **Request-URI** or **Content-Type** settings of the HTTP profile.

If you want to enable HTTP compression for specific connections, you can write an iRule that specifies the **HTTP:compress enable** command. For more information, see Chapter 13, *Writing iRules*.

When HTTP compression is enabled on the LTM system, the LTM system executes a series of steps:

1. First, the LTM system reads the **Accept-Encoding** header of a client request, looks for specification of either the **deflate** or **gzip** compression method, and notes whether either method is marked as being preferred.
2. If the **Keep Accept Encoding** setting in the HTTP profile is set to **Disabled**, the LTM system then removes the **Accept-Encoding** header from the request and passes the request on to the server.

Removing the **Accept-Encoding** header prevents the server from performing the HTTP compression and from inserting the **Content-Encoding** header into its response.

3. Upon receiving the server response, the LTM system inserts the **Content-Encoding** header, specifying the compression method that it has chosen to use. The LTM system chooses a compression method by looking for the specification of either the **gzip** or **deflate** compression method in the **Accept-Encoding** header of the client request. If the client request does not specify a compression method, the LTM system uses the **deflate** method to compress the response data.
4. Finally, the LTM system then compresses the response and sends it to the client. The client then reads the **Content-Encoding** header in the response, determines the compression method used, and uncompresses the data accordingly.

Using the LTM system HTTP compression feature, you can include or exclude certain types of URIs or files that you specify. This is useful because some URI or file types might already be compressed. Using CPU resources to compress already-compressed data is not recommended because the cost of compressing the data usually outweighs the benefits. Examples of regular expressions that you might want to specify for exclusion are**.*\.pdf**, **.*\.gif**, or **.*\.html**.

Table 6.6 shows the compression settings that you can specify within an HTTP profile. Configuring these settings means either specifying a value where no default value exists, or changing a default value.

Setting	Description	Default Value
Compression	Enables or disables the HTTP compression feature.	**Disable**
URI Compression	Displays the settings for including or excluding certain **Request-URI** responses. Possible values are **URI List** or **Not Configured**.	**Not Configured**
URI List	If the **URI Compression** setting is set to **Enabled**, specifies the URI targeted for compression, as well as the types of responses to include for and exclude from URI compression.	No default value
Content Compression	Displays the settings for including or excluding certain **Content-Type** responses. Possible values are **Content List** or **Not Configured**.	**Not Configured**
Content List	If the **Content Compression** setting is set to **Enabled**, specifies the type of content targeted for compression, as well as the type of responses to include for or exclude from content compression.	In the **Include List** box, default values are: **text/ application/(xml|x-javascript)**
Preferred Method	Specifies the compression method that you want to use to compress the response. Possible values are **gzip** and **deflate**.	**gzip**

Table 6.6 *Configurable settings for HTTP compression*

Setting	Description	Default Value
Minimum Content Length	Specifies the minimum length in bytes of a server response that is acceptable for compressing that response. The length applies to content length only, not headers.	**1024**
Compression Buffer Size	Specifies the maximum number of compressed bytes that the LTM system buffers before deciding whether or not to insert a **Content-Length** header into the response that specifies the compressed size.	**4096**
gzip Compression Level	Specifies the amount and rate of compression.	**1 - Least Compression (Fastest)**
gzip Memory Level	Specifies the number of kilobytes of memory that the LTM system uses for internal compression buffers when compressing a server response.	**8**
gzip Window Size	Specifies the number of kilobytes in the window size that the LTM system uses when compressing a server response.	**16**
Vary Header	Enables or disables the insertion of a **Vary** header into cacheable server responses.	**Enabled**
HTTP/1.0 Requests	Enables or disables compression of responses to HTTP/1.0 client requests.	**Disabled**
Keep Accept Encoding	When enabled, allows the target server to perform the HTTP compression instead of the LTM system.	**Disabled**
Browser Workarounds	Implements browser workarounds.	Disabled (unchecked)
CPU Saver	Specifies, when checked (enabled), that the system monitors the percent CPU usage and adjusts compression rates automatically when the CPU usage reaches either the **CPU Saver High Threshold** or the **CPU Saver Low Threshold**.	Enabled (checked)
CPU Saver High Threshold	Specifies the amount of CPU usage that causes the system to change the amount of content being compressed, and the amount of compression being applied.	**90**
CPU Saver Low Threshold	Specifies the amount of CPU usage that causes the system to revert back to user-defined compression values.	**75**

Table 6.6 *Configurable settings for HTTP compression*

Before configuring an HTTP profile, it is helpful to have a description of the compression settings that you might want to change.

Enabling or disabling the compression feature

The **Compression** setting causes the LTM system to perform compression on server responses. Possible values are:

- **Enabled**
  This value causes the LTM system to compress, or refrain from compressing, HTTP server content for any responses matching the values that you specify in the **URI List** or **Content List** settings of the HTTP profile.
- **Disabled**
  When the **Compression** setting is set to **Disabled**, the LTM system does not perform HTTP compression.
- **Selective**
  This setting causes the LTM system to perform HTTP compression only when specified by an iRule containing the **HTTP::compress** command.

To enable HTTP compression, locate the **Compression** setting and select **Enabled**. If you are enabling compression through the use of the iRule command **HTTP::compress <enable>**, select **Selective**.

Using URI compression

If you enable compression, you probably do not want the LTM system to compress every kind of server response. Using the **URI Compression** setting, you can set its value to **URI List**, which instructs the LTM system to include in compression, or exclude from compression, certain responses that are specified in the URIs of client requests.

More specifically, you can type regular expressions to specify the types of server responses that you want the LTM system to include in, or exclude from, compression. For example, you can specify that you want the LTM system to compress all**.htm** responses by typing the regular expression**.*.htm**. The LTM system then compares that response type to the URI specified within each client request, and if the system finds a match, takes some action.

Any regular expression that you specify must be in Advanced Regular Expression (ARE) syntax.

To use the URI Compression feature, locate the **URI Compression** setting and select **URI List**. You can then use the **Include List** box or the **Exclude List** box to type your regular expressions. If you do not specify any list (either URI or content), the LTM system compresses all responses. For more information on content lists, see *Using content compression*, on page 6-15.

Including URI-specified responses in HTTP compression

When the **URI compression** setting is enabled, and you type one or more values in the **Include List**, the LTM system compresses only those responses that match the URI part of the client request line.

The values you specify in the **Include List** box are in the form of regular expressions. For example, if the **Include List** box contains the values**.*.txt**,**.*.htm**, and**.*.html**, and those expressions match URIs in client requests, then the LTM system compresses only the responses for those URIs that match the specified regular expressions.

To apply this setting successfully, the LTM system must find a match in at least one of the values specified in the **Include List** box. If the LTM system finds no match, then no response is compressed.

Excluding URI-specified responses in HTTP compression

When the **URI compression** setting is enabled, and you type one or more values in the **Exclude List**, the LTM system excludes from compression those responses that match the URI part of the client request line.

The values you specify in the **Exclude List** box are in the form of regular expressions. For example, if the **Exclude List** box contains the value**.*.pdf**, and that expression matches URIs in client requests, then the LTM system excludes from compression any**.pdf** responses for those URIs.

To apply this setting successfully, the LTM system must find a match in at least one of the values specified in the **Exclude List** box. If the LTM system finds no match, then no responses are excluded from compression.

Using content compression

If you enable compression, you probably do not want the LTM system to compress every kind of server response. Using the **Content Compression** setting, you can set its value to **Content List**, which instructs the LTM system to include in compression, or exclude from compression, certain responses that are specified in the **Content-Type** header of server responses.

More specifically, you can type regular expressions to specify the types of server responses that you want the LTM system to include in, or exclude from, compression. For example, you can specify that you want the LTM system to compress all**.htm** responses by typing the regular expression**.*.htm**. The LTM system then compares that response type to the value of the **Content-Type** header specified within each server response, and if the system finds a match, takes some action.

Any regular expression that you specify must be in Advanced Regular Expression (ARE) syntax.

To use the Content Compression feature, locate the **Content Compression** setting and select **Content List**. You can then use the **Include List** box or the **Exclude List** box to type your regular expressions. If you do not specify any list (either URI or content), the LTM system compresses all responses. For more information on content lists, see *Using URI compression*, on page 6-14.

Including Content-Type responses in HTTP compression

When compression is enabled and you specify one or more values in the **Include List** box, the LTM system includes only those responses that match the value of the server's **Content-Type** header. The value of this setting is a list of those header values. For example, if the **Include List** box contains the values **application/pdf** and **image/.***, then only responses of those content types are compressed.

To include all text types, you assign the value **text/.*** to this setting.

Excluding Content-Type responses in HTTP compression

When compression is enabled and you specify one or more values in the **Exclude List** box, the LTM system excludes those responses that match the value of the server's **Content-Type** header. The value of this setting is a list of those header values. For example, if the **Exclude List** box contains the values **application/pdf** and **image/.***, then responses of those content types are excluded from compression.

To exclude all text types, you assign the value **text/.*** to this setting.

Specifying a preferred compression method

Using the **Preferred Method** setting, you can specify the compression method that you want the BIG-IP system to use when compressing responses. The two possible values are **gzip** and **deflate**. The default value is **gzip**.

Specifying minimum content length for compression

When compression is enabled, the **Minimum Content Length** setting specifies the minimum length of a server response in uncompressed bytes that the LTM system requires for compressing that response. The LTM system finds the content length of a server response in the **Content-Length** header of the server response. Thus, if the content length specified in the response header is below the value assigned to the **Minimum Content Length** setting, the LTM system does not compress the response. The length in bytes applies to content length only, not headers.

For example, using the default value of **1024**, the LTM system compresses only those responses with HTTP content containing at least 1024 bytes.

Sometimes the **Content-Length** header does not indicate the content length of the response. In such cases, the LTM system compresses the response, regardless of size.

To specify a minimum content length, locate the **Minimum Content Length** setting, and type a numeric value.

Specifying the compression buffer size

When compression is enabled, the **Compression Buffer Size** setting specifies the maximum number of compressed bytes that the LTM system buffers before deciding whether or not to preserve a **Keep-Alive** connection and rewrite the **Content-Length** header.

For example, using the default value of **4096**, the LTM system buffers up to 4096 bytes of compressed data before deciding whether or not to preserve the connection and rewrite the **Content-Length** header.

The LTM system's decision to rewrite the **Content-Length** header depends on whether response chunking is enabled (using the **Response Chunking** profile setting). Table 6.7 shows the behavior of the LTM system with respect to compression buffer size and response chunking.

If the size of the compressed response is	And the compressed response is	Then the LTM system
Equal to or greater than the maximum buffer size	Chunked	Keeps the connection open (if the **Connection** header is not set to **Close**).
	Not chunked	Closes the connection by changing the value of the **Connection** header to **Close**.
Less than the maximum buffer size	Chunked	Does not insert a **Content-Length** header with a compressed response size.
	Not chunked	Inserts a **Content-Length** header with a compressed response size.

Table 6.7 *LTM system behavior based on maximum buffer size*

For more information, see *Specifying minimum content length for compression*, preceding.

To specify a compression buffer size, locate the **Compression Buffer Size** setting and type a numeric value.

Specifying a compression level

Using the **gzip Compression Level** setting, you can specify the extent to which data is compressed, as well as the compression rate. Possible values are **1 - Least Compression (Fastest)**, **9 - Most Compression (Slowest)**, and **Other**. The default compression level is **1 - Least Compression (Fastest)**.

You can specify any whole number ranging from **1** through **9**, with these results:

- Specifying a lower number causes data to be less compressed but at a higher performance rate. Thus, a value of **1** causes the least compression but the fastest performance.
- Specifying a higher number causes data to be more compressed but at a slower performance rate. Thus, a value of **9** (the highest possible value) causes the most compression, but the slowest performance.

WARNING

*Selecting any value other than **1 - Least Compression (Fastest)** can degrade system performance.*

Specifying a memory level for gzip compression

The **gzip Memory Level** setting specifies a value representing the amount of kilobytes of memory that the LTM system uses to compress data when using the **gzip** or **deflate** compression method. The value of the **gzip Memory Level** setting must be a power-of-2 integer, in bytes, ranging from **1** to **256**.

Generally, a higher value causes the LTM system to use more memory, but results in a faster and higher compression ratio. Conversely, a lower value causes the LTM system to use less memory, but results in a slower and lower compression ratio. The default value is **8**.

To specify a memory level, locate the **gzip Memory Level** setting and select a numeric value.

Specifying window size for gzip compression

The **gzip Window Size** setting specifies a value representing the number of kilobytes in window size that the LTM system uses when compressing a server response using the **gzip** or **deflate** compression method. The value of the **gzip Window Size** setting must be a power-of-2 integer, in bytes, ranging from **1** to **128**.

Generally, a higher value causes the LTM system to use more memory, but results in a higher compression ratio. Conversely, a lower value causes the LTM system to use less memory, but results in a lower compression ratio. The default value is **16**.

To specify a window size, locate the **gzip Window Size** setting and select a numeric value.

Enabling or disabling the Vary header

When compression is enabled, the **Vary Header** setting inserts the **Vary: Accept-Encoding** header into a compressed server response. If the **Vary** header already exists in the response, the LTM system appends the value **Accept-Encoding** to that header.

The reason for inserting the **Vary: Accept-Encoding** header into a server response is to follow a recommendation by RFC2616, which states that the **Vary** header should be inserted into any cacheable response that is subject to server-driven negotiation. Server responses that are subject to HTTP compression fall into this category.

If the **Vary Header** setting is disabled, the LTM system does not insert the **Vary** header into a server response.

To disable the Vary header, locate the **Vary Header** setting and clear the **Enabled** box.

Allowing compression for HTTP/1.0 requests

The **HTTP/1.0 Requests** setting is included for backward compatibility, allowing HTTP compression for responses to HTTP/1.0 client requests. The default value for this setting is **Disabled**.

If this setting is set to **Enabled**, the LTM system only compresses responses in either of the following cases:

- When the server responds with a **Connection: close** header
- When the response content is no greater than the value of the **Compression Buffer Size** setting

To enable compression for HTTP/1.0 requests, locate the **HTTP/1.0 Requests** setting and check the box.

Keeping the Accept-Encoding header

Normally, when you enable HTTP compression, the LTM system strips out the **Accept-Encoding** header from the HTTP request. This causes the LTM system to perform the HTTP compression instead of the target server.

By default, the **Keep Accept Encoding** setting is disabled. If you want to allow the target server instead of the LTM system to perform the HTTP compression, simply enable this setting.

Implementing browser workarounds

When you enable the **Browser Workarounds** setting, the system uses built-in workarounds for several common browser issues that occur when compressing content. More specifically, enabling this setting prevents the system from compressing server responses when any of these conditions exist:

- The client browser is Netscape version 4.0x.
- The client browser is Netscape version 4.x (that is, versions 4.10 and higher), and the **Content-Type** header of the server response is not set to **text/html** or **text/plain**.
- The client browser is Microsoft® Internet Explorer (any version), the **Content-Type** header of the server response is set to either **text/css** or **application/x-javascript**, and the client connection uses SSL.
- The client browser is Microsoft® Internet Explorer (any version), the **Content-Type** header of the server response is set to either **text/css** or **application/x-javascript**, and the **Cache-Control** header of the server response is set to **no-cache**.

The default setting is disabled (unchecked).

CPU Saver

When you enable the **CPU Saver** setting, the system monitors the percent CPU usage and adjusts compression rates automatically when the CPU usage reaches the percentage defined in either the **CPU Saver High Threshold** or the **CPU Saver Low Threshold**. The default setting is enabled (checked).

CPU Saver High Threshold

The **CPU Saver High Threshold** setting specifies the percent of CPU usage at which the system starts automatically decreasing the amount of content being compressed, as well as the amount of compression which the system is applying. The default setting is **90** percent.

CPU Saver Low Threshold

The **CPU Saver Low Threshold** specifies the percent CPU usage at which the system resumes content compression at the user-defined rates. The default setting is **75** percent.

Configuring the RAM Cache

This section describes how to configure the properties of the RAM Cache feature on the BIG-IP system. A ***RAM cache*** is a cache of HTTP objects stored in the BIG-IP system's RAM that are reused by subsequent connections to reduce the amount of load on the back-end servers.

Getting started with RAM caching

There are several concepts to consider before you configure the RAM cache on the BIG-IP system.

- When to use the RAM cache
- What items you can cache
- The cache mechanism

When to use the RAM Cache

The RAM cache feature provides the ability to reduce the traffic load to back-end servers. This ability is useful if an object on a site is under high demand, if the site has a large quantity of static content, or if the objects on the site are compressed.

- **High demand objects**
  This feature is useful if a site has periods of high demand for specific content. With RAM cache configured, the content server only has to serve the content to the BIG-IP system once per expiration period.
- **Static content**
  This feature is also useful if a site consists of a large quantity of static content such as CSS, javascript, or images and logos.
- **Content compression**
  For compressible data, the RAM cache can store data for clients that can accept compressed data. When used in conjunction with the compression feature on the BIG-IP system, the RAM cache takes stress off of the BIG-IP system and the content servers.

The items you can cache

The RAM Cache feature is fully compliant with the cache specifications described in RFC 2616, *Hypertext Transfer Protocol -- HTTP/1.1*. This means you can configure the RAM Cache feature to cache the following content types:

- 200, 203, 206, 300, 301, and 410 responses
- Responses to GET methods by default.
- Other HTTP methods for URIs specified in the URI Include list or specified in an iRule.

- Content based on the **User-Agent** and **Accept-Encoding** values. The RAM cache holds different content for **Vary** headers.

The items that the RAM cache does not cache are:

- Private data specified by cache control headers
- HEAD, PUT, DELETE, TRACE, and CONNECT methods, by default.

Understanding the RAM Cache mechanism

The default RAM cache configuration caches only the HTTP GET methods. You can use the RAM cache to cache both the GET and other methods, including non-HTTP methods, by specifying a URI in the URI Include list or writing an iRule. The cache uses the storage mechanism for cached content described in Table 6.8.

Action	Cached Content
Removed	All cookie headers are removed.
Modified	Hop-by-hop headers are modified when served. These headers include: **Connection**, **Keep-Alive**, and **Transfer Encoding**.
Added	A **Date** header is added that includes the current time on the BIG-IP system. An Age header is added that reflects the amount of time the item has been in the cache. Note that this setting is on by default in the profile. You can disable this setting by turning the **Insert Age Header** attribute **off** in the profile.
Unchanged	All other headers are stored as they are received.

Table 6.8 *RAM cache storage mechanism*

Clearing items from the RAM cache

The RAM cache removes the least frequently used items in the cache. This prevents stale items from taking up room in the cache when newer items are selected for caching. The cache also uses a scoring system to remove stale items after a period of time. When a cached item reaches its age limit, it is expired from the cache. You can use the HTTP profile attributes to control the size of each cache instance and how quickly items are expired from the cache. For more information about these attributes, see the following section, *Understanding RAM Cache settings*.

Understanding RAM Cache settings

To configure the RAM cache, you need to configure the RAM cache settings of the HTTP profile. These settings provide the ability to turn on the cache and fine tune it for a specific implementation. In addition to configuring the RAM cache objects in the HTTP profile, you may want to use **bigpipe** utility commands, configure bigdb variables, or create iRules. For information on bigdb keys related to the RAM Cache feature, see Appendix B of the ***Network and System Management Guide***. For information on creating iRules for RAM cache, see Chapter 13, *Writing iRules*, in this guide.

The default items stored by the cache are HTTP GET responses. However, you can specify URIs in the URI list if you want to cache POST and GET methods for a particular URI.

The RAM cache settings are available in the HTTP profile. Table 6.9 lists and describes the RAM cache settings for the HTTP profile type.

Setting	Description	Default Value
RAM cache	Specifies whether the RAM Cache feature is enabled or disabled.	**Disabled**
Maximum Cache Size	Specifies the maximum size in megabytes that you want to allow for the RAM cache. When the cache reaches the maximum size, the system starts removing the oldest entries. **Note:** We recommend that you specify a value lower than the highest value allowed (on certain platforms, this is **448**). Setting the value too high automatically disables the RAM Cache feature and logs a TMM allocation error.	**100**
Maximum Entries	Specifies the maximum number of entries that you want to allow in the RAM cache. The default value of **0** means that the system does not limit the maximum number of entries. **Note:** We recommend that you specify a value lower than the highest value allowed. Setting the value too high automatically disables the RAM Cache feature.	**0**
Maximum Age	Specifies how long in seconds that the system considers the cached content to be valid.	**3600**
Minimum Object Size	Specifies the smallest object in bytes that the system considers eligible for caching.	**500**
Maximum Object Size	Specifies the largest object in bytes that the system considers eligible for caching.	**50000**
URI Caching	Specifies whether the system retains or excludes certain URIs in the RAM cache. The process forces the system either to cache URIs that typically are ineligible for caching, or to not cache URIs that typically are eligible for caching.	**Not Configured**

Table 6.9 *RAM cache settings in the HTTP profile*

Setting	Description	Default Value
URI List	Specifies the Uniform Resource Identifiers (URIs) that the system either includes in or excludes from caching. **Include List** Lists the URIs that are typically ineligible for caching, but the system caches them. When you add URIs to the include list, the GET methods are cached for the URI and other methods, including non-HTTP methods. **Exclude List** Lists the URIs that are typically eligible for caching, but the system does not cache them.	No default value
Ignore Headers	Specifies how the system processes client-side **Cache-Control** headers when RAM caching is enabled. **All** Specifies that the system disregards all **Cache-Control** headers. **Cache-Control:max-age** Specifies that the system ignores only the **Cache-Control:max-age** header. **None** Specifies that the system honors all **Cache-Control** headers.	**All**
Insert Age Header	Specifies, when enabled, that the system inserts **Date** and **Age** headers in the cached entry. The **Date** header contains the current date and time on the BIG-IP system. The **Age** header contains the length of time the content has been in the cache.	**Enabled**
Aging Rate	Specifies how quickly the system ages a cache entry. The aging rate ranges from 0 (slowest aging) to 10 (fastest aging).	**9**

Table 6.9 *RAM cache settings in the HTTP profile*

Configuring FTP profile properties and settings

You can tailor FTP profile properties and settings to your specific needs. For those settings that have default values, you can retain those default settings or modify them. You can modify any settings either when you create the profile, or at any time after you have created it. For specific procedures on configuring a profile, see Chapter 5, *Understanding Profiles*.

Table 6.10 lists these configurable settings, along with a short description of each and the default values. Following this table are descriptions of specific settings.

General property	Description	Default Value
Name	Specifies the user-supplied name of the profile. Specifying a name for your profile is required.	No default value
Parent Profile	Specifies the profile from which your custom profile is derived.	**FTP**
Translate Extended	Ensures compatibility between IP version 4 and IP version 6 clients and servers when using the FTP protocol.	**Enabled**
Data Port	Allows the FTP service to run on an alternate port.	**20**

Table 6.10 *Properties and settings of an FTP profile*

Before configuring an FTP profile, it is helpful to have a description of certain node settings that you might want to change.

Specifying a profile name

To create an FTP profile, you must specify a unique name for the profile. The **Name** setting is one of only two settings for which you must actively specify a value when creating an FTP profile; all other settings have default values.

Specifying a parent profile

Every profile that you create is derived from a parent profile. In the Parent Profile setting, you can select the default FTP profile as the parent profile, or you can select another FTP profile that you have already created.

Specifying a Translate Extended value

Because IP version 6 addresses are not limited to 32 bits (unlike IP version 4 addresses), compatibility issues can arise when using FTP in mixed IP-version configurations.

Enabled by default, the **Translate Extended** setting causes the LTM system to automatically translate FTP commands when a client-server configuration contains both IP version 4 and IP version 6 systems. For example, if a client system running IP version 4 sends the FTP **PASV** command to a server running IP version 6, the LTM system automatically translates the **PASV** command to the equivalent FTP command for IP version 6 systems, **EPSV**.

The LTM system translates the FTP commands **EPRV** and **PORT** in the way.

It is highly unlikely that you will need to change the default value (**Enabled**) for this setting. The only case where you might want to disable this setting is when sending an **EPSV** command to an IP version 4 system, such as when testing an FTP server.

Specifying a data port

The Data Port setting allows the FTP service to run on an alternate port. You can use the default port number **20**, or specify another port number.

Managing HTTP and FTP profiles

Using the Configuration utility, you can view HTTP and FTP profiles and delete any custom profiles that you have created. Note, however, that you cannot delete the default profiles **http** and **ftp**.

For the procedures on creating and modifying SSL profiles, see Chapter 5, *Understanding Profiles*.

To view an HTTP or FTP profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the **Services** menu, choose **HTTP** or **FTP**.
   This displays a list of any existing HTTP or FTP profiles.
4. In the Name column, click the name of the profile that you want to view.

Tip

*When listing existing profiles, you can use the **Search** box that appears directly above the profile list. With the **Search** box, you can specify a string to filter the list, thereby showing only those objects that match the string. The default setting is an asterisk (*), which means show all objects.*

To delete an HTTP or FTP profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the **Services** menu, choose **HTTP** or **FTP**.
   This displays a list of any existing client-side or server-side profiles.
4. Click the Select box to the left of the custom profile that you want to delete.
5. Click **Delete**.
   A confirmation screen appears.
6. Click **Delete**.

7

Managing SSL Traffic

- Introducing SSL traffic management
- Managing keys and certificates
- Understanding SSL profiles
- Configuring general properties of an SSL profile
- Configuring configuration settings
- Configuring client or server authentication settings
- Managing SSL profiles

Introducing SSL traffic management

The BIG-IP® local traffic management (LTM) system offers several features that you can use to intelligently control your SSL traffic. Some of the SSL traffic-management features are:

- The ability to authenticate clients and servers to maintain secure connections between a client system and the BIG-IP system, and between the BIG-IP system and a target web server
- The ability to offload SSL certificate-verification tasks from client and server systems
- iRule commands that provide features such as header insertion and HTTP redirection.
- Support for Online Certificate Status Protocol (OCSP), for the purpose of obtaining up-to-date certificate revocation status
- Certificate-based client authorization using a Light-weight Directory Access Protocol (LDAP) server

The primary way that you can control SSL network traffic is by configuring a Client or Server SSL profile. A ***Client profile*** is a type of traffic profile that enables the LTM system to accept and terminate any client requests that are sent by way of a fully SSL-encapsulated protocol. As an option, using a ***Server profile***, you can also configure an SSL profile to initiate secure connections to a target web server.

Managing SSL traffic requires you to complete these tasks:

- Install a key/certificate pair on the LTM system for terminating client-side secure connections.
- Configure a profile, and then associate that profile with a virtual server. You configure a profile using the Configuration utility.
- Associate the profile with a virtual server.

As an option, you can write an iRule that gives you the ability to manage individual SSL connections in certain ways. For more information, see Chapter 13, *Writing iRules*.

Managing client-side and server-side traffic

With the LTM system, you can enable SSL traffic management for either client-side traffic or server-side traffic. ***Client-side traffic*** refers to connections between a client system and the LTM system. ***Server-side traffic*** refers to connections between the LTM system and a target server system.

- **Managing client-side SSL traffic**
  When you enable the LTM system to manage client-side SSL traffic, the LTM system terminates incoming SSL connections by decrypting the client request. The LTM system then sends the request, in clear text, to a target server. Next, the LTM system 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 LTM system can, as an option, perform all of the SSL certificate verification functions normally handled by the target web server. For information on configuring a client-side SSL profile, see *Understanding SSL profiles*, on page 7-12.

- **Managing server-side SSL traffic**
  When you enable the LTM system to manage server-side SSL traffic, the LTM system enhances the security of your network by re-encrypting a decrypted request before sending it on to a target server. In addition to this re-encryption, the LTM system can, as an option, perform the same verification functions for server certificates that the LTM system can for client certificates. For information on configuring a server-side SSL profile, see *Understanding SSL profiles*, on page 7-12**.**

Summarizing SSL traffic-control features

The LTM system includes several important features related to controlling SSL traffic. Foremost among these features are SSL authentication (that is, certificate verification and revocation), as well as encryption and decryption. While most of these features are available through configuration of a Client or Server SSL profile, others are offered through features such as iRules™. Table 7.1 summarizes the features related to SSL traffic management; the sections following the table describe these features.

Features	Description	Configuration Tool
Certificate verification	In terminating and initiating SSL connections, the LTM system can perform a full range of certificate verification tasks, including the verification of both client and server certificates. For example, you can define the extent to which the system verifies client certificates: you can configure the system to request client certificates, require them, or ignore them altogether. You can also specify settings such as the depth of certificate chain traversal.	Client and Server SSL profiles iRules
Certificate revocation	When a client or server presents a certificate to the LTM system, the system can check the revocation status of a certificate as part of the certificate verification process.	Client and Server SSL profiles OCSP profile
Encryption and decryption	As a way to off-load work from target web servers, the LTM system decrypts incoming client requests. As an added security option, you can configure the profile 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 SSL profile configuration.	Client and Server SSL profiles

Table 7.1 *Summary of the LTM system features related to SSL traffic control*

Features	Description	Configuration Tool
Authorization	You can control access to system resources by configuring the LTM system in certain ways. For example, you can create an iRule 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 LTM system can grant access to resources by querying the LDAP server using client certificate credentials. You can also limit the number of concurrent TCP connections.	SSL Client Certificate LDAP profile iRules
SSL session persistence	A powerful feature of the LTM system is its ability to enable persistence for SSL sessions. Two types of SSL persistence are available, depending on whether you have configured the LTM system to terminate SSL connections.	Client and Server SSL profiles SSL Persistence profile iRules
Other SSL features	In addition to the features listed above, the LTM system 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.	Client and Server SSL profiles

Table 7.1 *Summary of the LTM system features related to SSL traffic control*

Understanding 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, the LTM system 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. As an option, you can configure the LTM system to handle the task of verifying client certificates.

When you configure certificate verification on the LTM system, the LTM system must hold a certificate that it can present to clients when processing SSL requests.

Optionally, you can configure the LTM system to verify server responses. In this case, if the server specifically requests a certificate from the LTM system, the LTM system needs to hold a second certificate.

If you configure the LTM system to verify client and server certificates, you must generate and install keys and certificates onto the LTM system before the system can manage SSL traffic. You can do this through the

Configuration utility. As an alternative to generating new key/certificate pairs, you can import existing ones. For more information on importing existing key/certificate pairs, see *Importing keys, certificates, and archives*, on page 7-10.

Client-side certificate verification

When a client makes an HTTP request, the LTM system can perform the client certificate verification task that is normally performed by the target server.

When a client presents a certificate to the LTM system, the LTM system uses a ***client trusted CAs file*** to determine the Certificate Authorities that it can trust. Using this file is the primary way that the LTM system attempts to verify a client certificate. When you create a client-side profile, the LTM system automatically creates a default client trusted CAs file. You can either use the default file name specified in the profile, or you can specify a different file name. For more information, see *Specifying trusted client CAs*, on page 7-16.

A second factor in client certificate verification is the ***client certificate CA file***, which the LTM system uses to advertise to clients a list of the CAs that the profile trusts. Not that this list could be different from the list of CAs that the LTM system *actually* trusts. As with the trusted CAs file, the LTM system automatically creates a a default version of this file.

There is also a client chain file, which the LTM system sends to a client as part of the client certificate verification process. The default client chain file is the Client Trusted CAs file. For more information, see *Specifying trusted client CAs*, on page 7-16.

Server-side certificate verification

Server-side verification occurs when you enable a Server SSL profile and set the presentation of a server certificate to **require**.

When a server presents a certificate to the LTM system, the LTM system uses the ***server trusted CAs*** file to determine which Certificate Authorities it can trust. Using this file is the primary way that the LTM system attempts to verify a server certificate. The LTM system automatically creates a default Server Trusted CAs file when you configure a server-side profile. You can either use the default file name specified in the profile, or specify a different file name.

There is also a server chain file, which the LTM system sends to a server as part of the entire server certificate verification process. The default server chain file is the Server Trusted CAs file.

Understanding certificate revocation

The LTM system can check to see if a certificate being presented by a client or server has been revoked. A revoked client certificate indicates to the LTM system that the system should fail to authenticate the client.

The LTM system supports two industry-standard methods for checking the revocation status of a certificate. These two methods are:

- **Certificate revocation lists (CRLs).**
  CRLs are a method that the LTM system can use to check on whether a certificate being presented to the LTM system has been revoked. This CRL support is in the form of a CRL file and a CRL path. The LTM system enables you to configure one CRL file and path for the client-side profile, and one CRL file and path for the server-side profile. You configure the use of CRLs through an SSL profile. For more information, see *Configuring client or server authentication settings*, on page 7-23.
- **Online Certificate Status Protocol (OCSP)**
  Unlike the use of CRLs, OCSP ensures that the revocation status of a certificate is always up-to-date. You configure OCSP through an Authentication profile. For more information, see Chapter 8, *Authenticating Application Traffic*.

Understanding encryption/decryption

Another feature of the LTM system is its ability to handle encryption and decryption tasks that a target server normally performs as part of processing a client request. Using a client-side SSL profile, the LTM system decrypts incoming requests before sending them on in plain text to the target server. Using a server-side profile, the LTM system provides an additional level of security by re-encrypting the request before sending it on to the target server.

An LTM system feature related to encryption is the ability to insert into incoming HTTP requests the cipher that the client used to encrypt its request. You can then direct the traffic based on that cipher specification. For more information on specifying ciphers to control the destination of client requests, see Chapter 13, *Writing iRules*.

Understanding client authorization

Unlike authentication, ***authorization*** is not about trusting identity, but about controlling access to system resources. Once the SSL profile has verified that a client or server can be trusted, the LTM system can then control the connection's level and type of access to the destination content.

The LTM system features 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

One of the most useful ways to control a client's access to system resources is to create an iRule that inserts fields of a client certificate as headers into client requests. For example, an iRule can insert the status of a client certificate as a header into a request, and then use the **HTTP::header** command to select the target server based on that status.

For more information on using this header insertion feature to control the destination of client requests, see Chapter 13, *Writing iRules*.

Understanding SSL session persistence

There are two types of SSL persistence available that you can implement. The first type is the standard SSL persistence mode, which enables persistence for SSL sessions that do not involve the decryption of SSL requests and the re-encryption of SSL responses. You enable this SSL persistence mode by configuring an SSL persistence profile. For more information, see Chapter 9, *Enabling Session Persistence*.

The second type of SSL persistence enables persistence for SSL sessions that involve the decryption of requests and re-encryption of responses. In this case, you implement SSL persistence is by inserting SSL session IDs as headers into HTTP requests. You insert session ID headers by writing an iRule. For information on iRules, see Chapter 13, *Writing iRules*.

Understanding other SSL features

In addition to using the preceding features, you can also perform tasks such as specifying invalid protocol versions, configuring the size and timeout values of the SSL session cache, and configuring SSL shutdown behavior. You can also configure an SSL profile to renegotiate SSL sessions based on timeout values and amount of data transmitted.

Managing keys and certificates

Before you can enable decryption and encryption of SSL connections, you must install one or more SSL certificates on the LTM system. The purpose of these certificates is described in the section *Understanding certificate verification*, on page 7-3.

To ease the task of generating certificate requests and sending them to certificate authorities, the LTM system provides a set of key management screens within the Configuration utility. You access these certificate management screens from the Local Traffic section of the Main tab.

When managing keys, you can:

- Display information about all existing key pairs and certificates.
- Create requests for new key pairs and certificates and submit those requests to certificate authorities.
- Renew certificate requests.
- Display key and certificate properties.
- Import and export PEM-formatted keys and certificates.

Displaying information about existing keys and certificates

Summary information about existing key pairs and certificates is available from the Configuration utility. The SSL Certificates screen displays the following information:

- Status of the certificate (valid, about to expire, or expired)
- The unique name that you assigned the certificate when creating the request
- The issuer of the certificate
- The expiration date

To display information about existing certificates and keys

1. On the Main tab, expand **Local Traffic**.
2. Click **SSL Certificates**.
   This opens the SSL Certificates screen and lists all certificates installed on the LTM system.
3. Click a certificate name.
   This displays the properties of that certificate.
4. If you want to see information about the key that is associated with that certificate, click **Key** on the menu bar.
   This displays the type and size of the key.

Creating a request for a new certificate and key

Using the Configuration utility, you can either generate a self-signed certificate (usually used for internal test purposes only) or you can create a request for a certificate/key pair, to be sent to a certificate authority. When you send a request to a certificate authority, the certificate authority returns a signed certificate.

You can send a certificate request to a certificate authority in either of two ways:

- You can copy the text of the newly-generated request from the Configuration utility screen and give it to the certificate authority (using cut and paste).
- You can download the newly-generated request to a file and transmit the file to the certificate authority.

The way to transmit the request to a certificate authority (either through pasting the text or through a file attachment) is by accessing the certificate authority's web site. The Configuration utility screen for submitting a request for signature by a certificate authority includes links to various certificate authority web sites.

To request a self-signed certificate

1. On the Main tab, expand **Local Traffic**.
2. Click **SSL Certificates**.
   This displays the SSL Certificates screen.
3. On the upper-right portion of the screen, click **Create**.
4. Type a unique name for the certificate.
5. For the **Issuer** setting, select **Self**.
6. Configure the **Common Name** setting, and any other settings you want.
7. In the Key Properties section, select a key size.
8. Click **Finished**.

To request a certificate from a certificate authority

1. On the Main tab, expand **Local Traffic**.
2. Click **SSL Certificates**.
   This displays the SSL Certificates screen.
3. On the upper-right portion of the screen, click **Create**.
4. Type a unique name for the certificate.
5. In the **Issuer** box, select **Certificate Authority**.
6. Configure the **Common Name** setting, and any other settings you want.

7. Click **Finished**.
   This displays your certificate request.
8. To download the request into a file on your system, either:
   - Copy the certificate from the **Request Text** box.
   - Click the button in the **Request File** box.
9. In the **Certificate Authorities** box, click a certificate authority name.
   This displays the web site for the certificate authority.
10. Follow the instructions on the web site for either pasting the copied request or attaching the generated request file.
11. Click **Finished**.

Renewing a certificate

All certificates include expiration dates. When a certificate expires, you must renew that certificate if you want to continue using it.

To renew a certificate

1. On the Main tab, expand **Local Traffic**.
2. Click **SSL Certificates.**
   This displays a list of existing certificates.
3. Click the name of the certificate that you want to renew.
   This displays the certificate properties.
4. At the bottom of the screen, click **Renew**.
5. Modify or retain settings.
   Note that a value for the **Common Name** setting is required.
6. Click **Finished**.

Deleting a certificate/key pair

You can delete certificates and keys that you no longer need.

To delete a certificate

1. On the Main tab, expand **Local Traffic**.
2. Click **SSL certificates**.
   This displays the list of existing certificates.
3. Check the Select box to the left of the name of the certificate you want to delete.

4. At the bottom of the screen, click **Delete**.
   A confirmation screen appears.
5. Click **Delete**.
   This removes the certificate.

Importing keys, certificates, and archives

If you have transferred a key/certificate pair, a certificate, or a key/certificate archive onto the LTM system from another system, and the certificate or archive is in the form of a file or a base-64 encoded text string, you can import this certificate or archive into the Configuration utility. By importing a certificate or archive into the Configuration utility, you ease the task of managing that certificate or archive. You can use the Import SSL Certificates and Keys screen only when the certificate you are importing is in Privacy Enhanced Mail (PEM) format.

To import a key pair, certificate, or archive

1. On the Main tab, expand **Local Traffic**.
2. Click **SSL Certificates**.
   This displays the list of existing certificates.
3. In the upper right corner of the screen, click **Import**.
4. Select the type of import (**Key**, **Certificate**, or **Archive**).
5. Select the import method (text or file).
6. Type the name of the key, certificate, or archive.
7. Click **Import**.

Creating an archive

You can create an archive for one or more keys and certificates. You create an archive when you want to export multiple keys and certificates to another system.

To create an archive

1. On the Main tab, expand **Local Traffic**.
2. Click **SSL Certificates**.
   This displays the list of existing certificates.
3. Click **Archive**.
4. Type a name for your archive file.
   The file will have a **.tgz** extension.
5. In the Key List area:

a) Select a key that you want to archive.

b) Use the Move button (**<<**) to transfer the key name to the **Enabled** box.

6. In the Certificate List area:

   a) Select a certificate that you want to archive.

   b) Use the Move button (**<<**) to transfer the certificate name to the **Enabled** box.

7. Click the **locallb.sslcert.Generate&DownloadArchive** button.

8. Click **Finished**.

Understanding SSL profiles

As described in Chapter 5, *Understanding Profiles*, a ***profile*** is a group of settings with values that determine the way that the LTM system manages application-specific network traffic. One type of traffic that a profile can manage is SSL traffic. The most basic functions of an SSL profile are to offload certificate validation and verification tasks, as well as encryption and decryption, from your target web servers.

The two types of SSL profiles are:

- **Client profiles**
  Client profiles allow the LTM system to handle authentication and encryption tasks for any SSL connection coming into an LTM system from a client system. You implement this type of profile by using the default **clientssl** profile, or by creating a custom profile based on the **clientssl** profile.
- **Server profiles**
  Server profiles allow the LTM system to handle encryption tasks for any SSL connection being sent from a LTM system to a target server. A server SSL profile is able to act as a client by presenting certificate credentials to a server when authentication of the LTM system is required You implement this type of profile by using the default **serverssl** profile, or creating a custom profile based on the **serverssl** profile.

Important

Prior to enabling SSL processing, you must first generate either a valid x509 certificate from a Trusted Certificate Authority or a temporary certificate, and install it on the LTM system. For more information on certificates, see ***Signed certificates****, on page 7-3. For instructions on how to generate and install a certificate on the LTM system, see* ***Managing keys and certificates****, on page 7-7.*

The settings that make up a Client or Server SSL profile are organized into three categories: General properties, Configuration, and either Client Authentication or Server Authentication. You can configure these settings at the time that you create an SSL profile or after profile creation by modifying the profile's settings. For specific procedures on configuring a profile, see Chapter 5, *Understanding Profiles*.

Tip

For any individual SSL connection, you can override the value of an SSL profile setting. You do this by writing an iRule and including those SSL iRule commands that correspond to the settings you want to override. For more information, see Chapter 13, ***Writing iRules****.*

Configuring general properties of an SSL profile

This section describes the settings that appear in the General Properties section of a Client SSL Profile or Server SSL Profile screen. For information on other SSL profile settings, see *Configuring configuration settings*, on page 7-14 and *Configuring client or server authentication settings*, on page 7-23.

For the procedure on creating or modifying a profile, see Chapter 5, *Understanding Profiles*.

Table 7.2 shows the general properties that you can specify for a Client or Server SSL profile. For those settings that have default values, you can retain those default settings or modify them. Following this table are descriptions of these general properties.

General Property	Description	Default Value
Name	Specifies the user-supplied name of the profile.	No default value
Parent Profile	Specifies the system-supplied profile from which a custom profile is derived.	**clientssl** or **serverssl**

Table 7.2 *General Properties of an SSL profile*

When you configure the general properites of an HTTP profile, you specify a unique name for your profile and you select a parent profile.

Specifying a profile name

To create an SSL profile, you must specify a unique name for the profile. The **Name** setting is the only setting that you must actively specify when creating an SSL profile; all other settings have default values.

Selecting a parent profile

Every profile that you create is derived from a parent profile. Using the **Parent Profile** setting, you can configure the default SSL profile as the parent profile, or you can configure another SSL profile that you have already created.

Configuring configuration settings

This section describes the settings that appear in the Configuration section of a Client SSL Profile or Server SSL Profile screen. For information on configuring the other SSL profile settings, see *Configuring general properties of an SSL profile*, on page 7-13, and *Configuring client or server authentication settings*, on page 7-23.

For the procedure to create or modify an SSL profile, see Chapter 5, *Understanding Profiles*.

Table 7.3 shows the settings you can configure for a Client SSL or Server SSL profile. For those settings that have default values, you can retain those default settings or modify them. Following this table are descriptions of these settings.

Setting	Description	Default Value
Certificate	Specifies the name of the certificate installed on the LTM system for the purpose of terminating or initiating an SSL connection.	**default**
Key	Specifies the name of the key installed on the LTM system for the purpose of terminating or initiating an SSL connection.	**default**
Pass Phrase	Specifies the name of the pass phrase used to encrypt the key.	No default value
Confirm Pass Phrase	Confirms the name of the pass phrase used to encrypt the key.	No default value
Chain	Specifies or builds a certificate chain file that a client can use to authenticate the profile. For more information, see *Configuring a certificate chain*, on page 7-15.	No default value
Trusted Certificate Authorities	Configures certificate verification by specifying a list of client or server CAs that the LTM system trusts. For more information, see *Specifying trusted client CAs*, on page 7-16.	No default value
Ciphers	Specifies the list of ciphers that the LTM system supports. For more information, see *Specifying SSL ciphers*, on page 7-16.	**Default**
Options	Specifies the value **All Bugfixes Enabled**, which enables a set of industry-related miscellaneous workarounds related to SSL processing. For more information, see *Configuring workarounds*, on page 7-17.	**All Bugfixes Enabled**
Cache Size	Specifies the number of sessions in the SSL session cache. **Note**: This setting applies to client-side profiles only.	**20000**
Cache Timeout	Specifies the timeout value in seconds of the SSL session cache entries. **Note**: This setting applies to client-side profiles only.	300
Alert Timeout	Specifies the duration in seconds of the timeout of the SSL session cache.	Indefinite

Table 7.3 *Configurable settings for an SSL profile*

Setting	Description	Default Value
Renegotiate Period	Forces the specified period of SSL renegotiation.	**Indefinite**
Renegotiate Size	Forces the specified throughput size, in bytes, of SSL renegotiation.	**429496729**
Renegotiate Max Record Delay	Forces the maximum record delay of SSL renegotiation. **Note**: The **Renegotiate Max Record Delay** attribute applies to client-side profiles only.	**Indefinite**
Unclean Shutdown	Configures the LTM system to enable or disable unclean SSL shutdowns.	Checked
Strict Resume	Configures the LTM to enable or disable the resumption of SSL sessions after an unclean shutdown.	Unchecked

Table 7.3 *Configurable settings for an SSL profile*

Before configuring an SSL profile, it is helpful to have a description of certain settings that you might want to change.

Specifying a certificate name

The value of the **Certificate** setting is the name of the certificate that you installed on the LTM system for the purpose of authenticating client-side or server-side SSL connections. If you have not generated a certificate request nor installed a certificate on the LTM system, you can specify the name of the default certificate, **default**. For more information on certificates, see *Understanding certificate verification*, on page 7-3.

Specifying a key name

The value of the **Key** setting is the name of the key that you installed on the LTM system for the purpose of authenticating either client-side or server-side SSL connections. If you have not generated a key request nor installed a key on the LTM system, you can specify the name of the default key, **default**.

For more information on keys, see *Understanding certificate verification*, on page 7-3.

Configuring a certificate chain

In any client verification process, not only does the LTM system need to authenticate the client, but the client might need to authenticate the LTM system. However, a certificate that the LTM system 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 LTM system might need to use a certificate chain. The profile enables you to specify the name of a specific certificate chain file. Note that the certificate files that make up the chain file must be in PEM format.

Specifying trusted client CAs

For client-side SSL processing, you can configure an SSL profile to verify certificates presented by a client or a server. Using the **Trusted Certificate Authorities** setting, you can specify a client trusted CAs file name, which the LTM system then uses to verify client or server certificates. If you do not configure a trusted CAs file, the profile uses a default file.

The 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 or server certificates that the SSL profile will trust. If you do not specify a trusted CAs file, or the specified trusted CAs file is not accessible to the LTM system, the system uses the default file name.

Specifying SSL ciphers

For each SSL profile, you can specify the ciphers available for SSL connections. When configuring ciphers, you must ensure that the ciphers configured for the SSL profile match those of the client sending a request, or of the server sending a response.

For example, a client might connect to and successfully establish an SSL connection to an SSL profile 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 profile attempts to establish an SSL connection to a server. However, the SSL profile 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 profile 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 specify a string to indicate the available list of SSL ciphers, or you can use the default cipher string, **DEFAULT**. For Client SSL profiles, the definition of the **DEFAULT** cipher string is **ALL:!SSLv2:@SPEED**. For Server SSL profiles, the definition of the **DEFAULT** cipher string is **COMPAT+HW:@SPEED**.

The ciphers that the LTM system supports are:

- SSLv2
- SSLv3
- TLSv1
- SGC/Set-up

- All standard protocol extensions and ciphers described in RFC 2246
- AES ciphers (described in RFC 3268)

We do not recommend use of the **SSLv2** cipher unless necessary. Therefore, concerned users should set the cipher string to **DEFAULT:-SSLv2**.

Note

The LTM system supports the cipherlist format of OpenSSL version 0.9.7.

Tip

In addition to specifying ciphers in an SSL profile, you can insert cipher specifications into the header of an HTTP request and then direct traffic based on those ciphers. For more information, see Chapter 13, ***Writing iRules.***

Configuring workarounds

OpenSSL supports a set of defect workarounds and SSL options. You can enable these workarounds and options as settings of an individual client-side or server-side SSL profile. The default value for the **Options** setting is **All Options Disabled**. The value **ALL_BUGFIXES** enables all workarounds in the set.

Table 7.4 lists and describes the possible workarounds and options that you can configure for an SSL profile.

SSL Attribute	Description
All Bugfixes Enabled	This option enables all of the defect workarounds that this table describes. It is usually safe to use the **All Bugfixes Enabled** option to enable the defect workaround options when you want compatibility with broken implementations.
Cipher server preference	When the LTM system chooses a cipher, this option uses the server's preferences instead of the client preferences. When this option is not set, the SSL server always follows the client's preferences. When this option is set, the SSLv3/TLSv1 server chooses by using its own preferences. Due to the different protocol, for SSLv2 the server sends its list of preferences to the client, and the client always chooses the cipher.
Don't insert empty fragments	This option disables a countermeasure against a SSL 3.0/TLS 1.0 protocol vulnerability affecting CBC ciphers. These ciphers cannot be handled by certain broken SSL implementations. This option has no effect for connections using other ciphers.

Table 7.4 *Workarounds and other SSL options*

SSL Attribute	Description
Ephemeral RSA	This option uses ephemeral (temporary) RSA keys when doing RSA operations. According to the specifications, this is only done when an RSA key can only be used for signature operations (namely under export ciphers with restricted RSA key length). By setting this option, the LTM system always uses ephemeral RSA keys. This option breaks compatibility with the SSL/TLS specifications and can lead to interoperability problems with clients, and we therefore do not recommend it. You should use ciphers with EDH (ephemeral Diffie-Hellman) key exchange instead. This option is ignored for server-side SSL.
Microsoft session ID bug	This option handles a Microsoft session ID problem.
Netscape CA DN bug workaround	This option handles a defect regarding system instability. If the system accepts a Netscape browser connection, demands a client cert, has a non-self-signed CA that does not have its CA in Netscape, and the browser has a certificate, then the system crashes or hangs.
Netscape challenge bug	This option handles the Netscape challenge problem.
Netscape demo cipher change bug workaround	This option deliberately manipulates the SSL server session resumption behavior to mimic that of certain Netscape servers (see the **Netscape reuse cipher change bug workaround** description). We do not recommend this option for normal use and it is ignored for server-side SSL processing.
Netscape reuse cipher change bug workaround	This option handles a defect within Netscape-Enterprise/2.01 (**https://merchant.neape.com**), only appearing when connecting through SSLv2/v3 then reconnecting through SSLv3. In this case, the cipher list changes. First, a connection is established with the RC4-MD5 cipher list. If it is then resumed, the connection switches to using the DES-CBC3-SHA cipher list. However, according to RFC 2246, (section 7.4.1.3, cipher_suite) the cipher list should remain RC4-MD5. As a workaround, you can attempt to connect with a cipher list of DES-CBC-SHA:RC4-MD5 and so on. For some reason, each new connection uses the RC4-MD5 cipher list, but any re-connect ion attempts to use the DES-CBC-SHA cipher list. Thus Netscape, when reconnecting, always uses the first cipher in the cipher list.
No SSLv2	Do not use the SSLv2 protocol.
No SSLv3	Do not use the SSLv3 protocol.
No session resumption on renegotiation	When the LTM system performs renegotiation as an SSL server, this option always starts a new session (that is, session resumption requests are only accepted in the initial handshake). The system ignores this option for server-side SSL processing.
N0 TLSv1	Do not use the TLSv1 protocol.

Table 7.4 *Workarounds and other SSL options*

SSL Attribute	Description
Microsoft big SSLV3 buffer	This option enables a workaround for communicating with older Microsoft applications that use non-standard SSL record sizes.
Microsoft IE SSLV2 RSA padding	This option enables a workaround for communicating with older Microsoft applications that use non-standard RSA key padding. This option is ignored for server-side SSL.
Passive close	When set to **Default**, this option enables the **All Bugfixes Enabled** option. When set to **None**, this option disables all workarounds. We do not recommend setting this option to **None**.
PKCS1 check 1	This debugging option deliberately manipulates the PKCS1 padding used by SSL clients in an attempt to detect vulnerability to particular SSL server vulnerabilities. We do not recommend this option for normal use. The system ignores this option for client-side SSL processing.
PKCS1 check 2	This debugging option deliberately manipulates the PKCS1 padding used by SSL clients in an attempt to detect vulnerability to particular SSL server vulnerabilities. We do not recommend this option for normal use. The system ignores this option for client-side SSL processing.
Single DH use	This option creates a new key when using temporary/ephemeral DH parameters. You must use this option if you want to prevent small subgroup attacks, when the DH parameters were not generated using “strong” primes (for example, when using DSA-parameters). If “strong” primes were used, it is not strictly necessary to generate a new DH key during each handshake, but we do recommend this. You should enable the **Single DH use** option whenever temporary/ephemeral DH parameters are used.
SSLEAY 080 client DH bug workaround	This option enables a workaround for communicating with older SSLeay-based applications that specify an incorrect Diffie-Hellman public value length. This option is ignored for server-side SSL.
SSL Ref2 reuse cert type bug	This option handles the SSL reuse certificate type problem.
TLS D5 bug workaround	This option is a workaround for communicating with older TLSv1-enabled applications that specify an incorrect encrypted RSA key length. This option is ignored for server-side SSL.
TLS block padding bug workaround	This option enables a workaround for communicating with older TLSv1-enabled applications that use incorrect block padding.
TLS rollback bug workaround	This option disables version rollback attack detection. During the client key exchange, the client must send the same information about acceptable SSL/TLS protocol levels as it sends during the first **hello**. Some clients violate this rule by adapting to the server's answer. For example, the client sends an SSLv2 **hello** and accepts up to SSLv3.1 (TLSv1), but the server only understands up to SSLv3. In this case, the client must still use the same SSLv3.1 (TLSv1) announcement. Some clients step down to SSLv3 with respect to the server's answer and violate the version rollback protection. This option is ignored for server-side SSL.

Table 7.4 *Workarounds and other SSL options*

Note that when configuring protocol versions, you must ensure that the protocol versions configured for the LTM system match those of the system's peer. That is, protocol versions specified in the client-side SSL profile must match those of the client, and protocol versions specified in the server-side SSL profile must match those of the server. Thus, for both client-side and server-side SSL connections, you can specify the protocol versions that you do not want the LTM system to allow.

You can declare up to two of the three protocol versions to be invalid: **SSLv2**, **SSLv3**, and **TLSv1**. If no protocol versions are specified, the LTM system allows all SSL protocol versions.

Note

We recommend that at a minimum you specify protocol version SSLv2 as invalid.

To specify workaround options, locate the **Options** setting and select **Options List.** This displays the **Options List** setting. From the **Options List** setting, select any of the options you wish to configure, and click the **Enable** button.

Configuring the SSL session cache

For client-side profiles only, you can configure timeout and size values for the SSL session cache. Because each profile maintains a separate SSL session cache, you can configure the values on a per-profile basis.

Setting SSL session cache size

Using the Configuration utility, 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.

Note this setting applies to client-side profiles only.

You configure the client-side values for the maximum size of the session cache on a per-profile basis. To specify an SSL session cache size, locate the **Cache Size** setting and retain the default cache-size value or type a new value.

Setting SSL session cache timeout

Using the Configuration utility, 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.

Note this setting applies to client-side profiles only.

You configure the client-side values for the session cache timeout on a per-profile basis. To specify an SSL session cache timeout value, locate the **Cache Timeout** setting and retain the default cache timeout value or type a new value.

WARNING

If the timeout value for the client-side SSL session cache is set to zero, the SSL session IDs negotiated with that profile's clients remain in the session cache until 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.

To specify an SSL session cache timeout, locate the **Cache Timeout** setting, and retain the default value or, from the list, select **Specify, Immediate**, or **Indefinite**. If you select **Specify**, type a value.

Specifying an alert timeout

The **Alert Timeout** setting represents the duration in seconds of the timeout of entries in the SSL session cache.

Forcing renegotiation of SSL sessions

Long-lived connections are susceptible to man-in-the-middle attacks. To prevent such attacks, you can force the LTM system to renegotiate SSL sessions, based on either time period or application size. You can also force the LTM system to terminate an SSL session after receiving a specified number of records.

Renegotiating sessions based on a time period

The **Renegotiate Period** setting forces the LTM system to renegotiate an SSL session after the number of seconds that you specify.

To specify a renegotiation period, locate the **Renegotiate Period** setting and retain the **Indefinite** value or select **Specify**. If selecting **Specify**, type a value.

Renegotiating sessions based on application data size

The **Renegotiate Size** setting forces the LTM system to renegotiate an SSL session after the specified number of megabytes of application data have been transmitted over the secure channel.

To specify a renegotiation size, locate the **Renegotiate Size** setting and retain the default value or type a new value.

Specifying the maximum record delay

While the LTM system waits for the client to initiate a renegotiation, the **Renegotiate Max Record Delay** setting forces the LTM system to terminate an SSL session after receiving the specified maximum number of SSL records. If the LTM system receives more than the maximum number of SSL records, it closes the connection.

To specify a maximum record delay, locate the **Renegotiate Max Record Delay** setting and retain the **Indefinite** value or select **Specify**. If selecting **Specify**, type a value.

Configuring SSL shutdowns

With respect to the shutdown of SSL connections, you can configure two settings on the LTM system: **Unclean Shutdown** and **Strict Resume**.

Disabling unclean SSL shutdowns

In an ***unclean shutdown***, underlying TCP connections are closed without exchanging the required SSL shutdown alerts. However, you can disable unclean shutdowns and thus force the SSL profile to perform a clean shutdown of all SSL connections by configuring this 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 profile cannot always detect this requirement or lack of it. In the case where the browser expects a shutdown alert but the SSL profile has not exchanged one (the default setting), the browser displays an error message.

By default, the LTM system performs unclean shutdowns of all SSL connections. To disable unclean shutdowns, locate the **Unclean Shutdown** setting and clear the check box.

Discontinuing SSL sessions

You can configure the LTM system to discontinue an SSL session after an unclean shutdown. By default, this setting is disabled, which causes the LTM system to resume SSL sessions after an unclean shutdown. If you enable this setting, the LTM system does not resume SSL sessions after an unclean shutdown.

To discontinue SSL sessions after an unclean shutdown, locate the **Strict Resume** setting and check the box.

Configuring client or server authentication settings

This section describes the settings that appear in the Client Authentication section of a Client SSL Profile screen, or the Server Authentication section of a Server SSL Profile screen. For information on configuring the other SSL profile settings, see *Configuring general properties of an SSL profile*, on page 7-13, and *Configuring configuration settings*, on page 7-14.

For the procedure to create or modify an SSL profile, see Chapter 5, *Understanding Profiles*.

Table 7.5 lists and describes the authentication settings of a Client or Server SSL profile. For those settings that have default values, you can retain those default settings or modify them. Following this table are descriptions of the settings.

Setting	Description	Default Value
Client or Server Certificate	Configures the SSL profile to either request, require, or ignore certificates presented by a client or a server.	**Ignore**
Frequency	Specifies whether the profile should authenticate a client once per session, or once per session and upon each subsequent reuse of an SSL session. For more information, see *Configuring per-session authentication*, on page 7-25.	**Once**
Advertised Trusted Certificate Authorities	Specifies the CAs that you would like to advertise to clients as being trusted by the profile. For more information, see *Advertising a list of trusted client CAs*, on page 7-25. **Note**: This attribute applies to client-side profiles only.	No default value
Certificate Chain Traversal Depth	Specifies the maximum number of certificates that can be traversed in a client certificate chain. For more information, see *Configuring authentication depth*, on page 7-26.	**9**
Authenticate Name	Authenticates a target server based on the Common Name (CN) embedded in a server certificate. For more information, see *Configuring name-based authentication*, on page 7-26. **Note**: This attribute applies to server-side profiles only.	No default value
Certificate Revocation List (CRL)	Configures certificate revocation by maintaining a list of revoked client certificates. For more information, see *Certificate revocation*, on page 7-26.	No default values

Table 7.5 *Authentication settings in an SSL profile*

Configuring certificate presentation

By configuring the **Client Certificate** or **Server Certificate** setting, you can cause the LTM system to handle authentication of clients or servers in certain ways.

For client-side processing, the possible values of the **Client Certificate** setting are:

- **Request**
  Request and verify a client certificate. In this case, the SSL profile always grants access regardless of the status or absence of the certificate.
- **Require**
  Require a client to present a valid and trusted certificate before granting access.
- **Ignore**
  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 7-25.
- **Auto**
  Ignore a client certificate until an authentication module requests one. In this case, the LTM system initiates a mid-session SSL handshake, as though the option were set to **request**. We recommend this setting only for those connections for which the presentation of client certificate is not required.

WARNING

*If you 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 on LDAP-based client authorization, see Chapter 8, **Authentication Application Traffic**.*

Tip

*The **request** option works well with the header insertion feature. Configuring the SSL profile to insert client certificate information into an HTTP client request, and to authenticate clients based on the **request** option, enables the LTM 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.*

For server-side processing, the possible values of the **Server Certificate** setting are:

- **Require**
  Require a server to present a valid and trusted certificate before granting access.
- **Ignore**
  Ignore a certificate (or lack of one) and therefore never authenticate the server. 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 7-25.

Configuring per-session authentication

With the **Frequency** setting, you can configure an SSL profile 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.

You can modify the SSL profile to require authentication not only once per session, but also upon each subsequent reuse of an SSL session.

Advertising a list of trusted client CAs

For client-side profiles only, if you intend to configure the SSL profile to require or request client certificates for authentication, you will want the profile to send to clients a list of CAs that the server is likely to trust. You can do this by configuring the **Advertised Trusted Certificate Authorities** setting.

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 profile trusts. The client certificate CA file solves this problem by allowing the profile 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 browsers 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 profile 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.

Configuring authentication depth

Using the **Certificate Chain Traversal Depth** setting, you can configure the maximum number of certificates that can be traversed in the 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 or server certificate verification fails. If the authentication depth value is set to zero, then only the client or server certificate, and one of the chain files, are examined.

Configuring name-based authentication

For server-side profiles only, the LTM system supports name-based authentication, which guards against man-in-the-middle attacks. When you configure the **Authenticate Name** setting for a server-side profile, the LTM system checks the name against the Common Name (CN) listed in the certificate that the target server presents to the LTM system. If the name attribute that you specify does not match the CN in the server certificate, the LTM system closes the connection. An example of a CN is **www.f5.com**.

Certificate revocation

The **Certificate Revocation List (CRL)** setting allows the LTM system to use CRLs to check revocation status of a certificate prior to authenticating a client or server.

To configure CRLs for an SSL profile, you must configure a CRL file, which contains a list of revoked client or server certificates. When specifying a list of revoked certificates, the file that you specify must be a PEM-formatted file.

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 LTM 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/ssl/ssl.crl*** *directory:* ***openssl crl -in <crlname> -text -noout****.*

As an alternative to using CRLs, you can use the Online Certificate Status Protocol (OCSP) feature, which ensures up-to-date information on certificate revocation status. For more information, see Chapter 8, *Authenticating Application Traffic*.

Managing SSL profiles

Using the Configuration utility, you can view SSL profiles and delete any custom profiles that you have created. Note, however, that you cannot delete the default profiles **clientssl** and **serverssl**.

For the procedures on creating and modifying SSL profiles, see Chapter 5, *Understanding Profiles*.

To view or delete an SSL profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the SSL menu, choose Client or Server.
   This displays a list of any existing client-side or server-side profiles.
4. To delete a profile:
   a) Click the box to the left of the custom profile that you want to delete.
   b) Click **Delete**.
      A confirmation screen appears.
5. Click **Delete**.
   This removes the profile.
6. To view details of an SSL profile, in the Name column, click the name of the profile that you want to view.
   This displays the profile details.

◆ Tip

*When listing existing profiles, you can use the **Search** box that appears directly above the profile list. With the **Search** box, you can specify a string to filter the list, thereby showing only those objects that match the string. The default setting is an asterisk (*), which means show all objects.*

8

Authenticating Application Traffic

- Introducing remote authentication
- Implementing an LDAP authentication module
- Implementing a RADIUS authentication module
- Implementing a TACACS+ authentication module
- Implementing an SSL client certificate LDAP authentication module
- Implementing an SSL OCSP authentication module

Introducing remote authentication

A significant feature of the BIG-IP® local traffic management (LTM) system is its ability to support Pluggable Authentication Module (PAM) technology. PAM technology allows you to choose from a number of different authentication and authorization schemes to use to authenticate or authorize network traffic.

The goal of PAM technology is to separate an application, such as the LTM system, from its underlying authentication technology. This means that you can dictate the particular authentication/authorization technology that you want the LTM system to use to authenticate application traffic coming into the BIG-IP system.

To this end, the LTM system offers several authentication schemes, known as authentication modules. These ***authentication modules*** allow you to use a remote system to authenticate or authorize application requests that pass through the LTM system.

To implement an authentication module, you use the Configuration utility to access the profile screens within the Local Traffic section of the utility. Using these screens, you can configure settings for the type of authentication module you want to implement.

◆ Note

The BIG-IP system normally routes remote authentication traffic through a Traffic Management Microkernel (TMM) switch interface (that is, an interface associated with a VLAN and a self IP address), rather than through the management interface. Therefore, if the TMM service has been stopped for any reason, remote authentication is not available until the service is running again. For more information on routing traffic through BIG-IP interfaces, see the ***Network and System Management Guide****.*

LTM authentication modules

The LTM modules that you can implement for remote authentication are:

- **Lightweight Directory Access Protocol (LDAP)**
  The LTM system can authenticate or authorize network traffic using data stored on a remote LDAP server or a Microsoft® Windows Active Directory server. Client credentials are based on basic HTTP authentication (user name and password).
- **Remote Authentication Dial-In User Service (RADIUS)**
  The LTM system can authenticate network traffic using data stored on a remote RADIUS server. Client credentials are based on basic HTTP authentication (user name and password).

- **TACACS+**
  The LTM system can authenticate network traffic using data stored on a remote TACACS+ server. Client credentials are based on basic HTTP authentication (user name and password).
- **SSL client certificate LDAP**
  The LTM system can authorize network traffic using data stored on a remote LDAP server. Client credentials are based on SSL certificates, as well as defined user groups and roles.
- **Online Certificate Status Protocol (OCSP)**
  The LTM system can check on the revocation status of a client certificate using data stored on a remote OCSP server. Client credentials are based on SSL certificates.

Implementing authentication modules

Using the LTM system, you can implement any of the available authentication modules listed in the previous section. Implementing an authentication module requires you to create or configure the following objects:

- **A RADIUS server or SSL OCSP responder object**
  This is required for RADIUS and SSL OCSP authentication modules only. S***erver objects*** and ***responder objects*** consist of settings pertaining to remote RADIUS servers or OCSP responders.
- **A configuration object**
  It is the ***configuration object*** that controls the authentication module, through a group of configurable settings. Examples of configuration object settings are **Hosts**, **Service Port**, and **Search Time Limit**.
- **An authentication profile**
  An ***authentication profile*** is an object that specifies: the type of authentication module you want to implement, a parent profile, and the configuration object (see the previous step). You can either use the default profile that the LTM system provides for each type of authentication module, or create a custom profile. For background information on profiles, see Chapter 5, *Understanding Profiles*.
- **An authentication iRule**
  For any type of PAM module that you want to implement, the LTM system provides a corresponding default authentication iRule that you must associate with your profile and your virtual server. An ***authentication iRule*** is a TCL-based script that performs the authentication/authorization action, according to the settings of the corresponding profile and configuration objects.

  In most cases, you can use the default iRule, rather than creating a custom one. Creating custom iRules™ is necessary when you want to implement multiple authentication modules of the same type, such as

multiple LDAP authentication modules, each with different configuration object and profile settings. For information on creating custom iRules, see Chapter 13, *Writing iRules*.

The process you use to implement an authentication module is a simple one. For example, to implement an LDAP authentication module, which uses a remote LDAP server to authentication client traffic using user name and password credentials, you perform the following tasks. Note that this example creates a custom profile and uses the default iRule that the LTM system provides.

To implement an authentication module

1. Create an authentication configuration object named **my_ldap_config**. For information on creating an LDAP authentication configuration object, see *Creating an LDAP configuration object*, on page 8-4.
2. Create a custom authentication profile named **my_ldap_profile**, in which you specify the authentication module type as **LDAP**, specify a parent profile (either the default **ldap** profile or another custom profile that you created), and reference the configuration object **my_ldap_config**. For information on creating an LDAP profile, see *Creating an LDAP profile*, on page 8-6.
3. Configure the virtual server to reference both the custom profile **my_ldap_profile** and the default authentication iRule **auth_ldap**. For information on how to configure virtual server settings, see Chapter 2, *Configuring Virtual Servers*.

Implementing an LDAP authentication module

An LDAP authentication module is a mechanism for authenticating or authorizing client connections passing through an LTM system. This module is useful when your authentication or authorization data is stored on a remote LDAP server or a Microsoft® Windows Active Directory server, and you want the client credentials to be based on basic HTTP authentication (that is, user name and password).

To implement an LDAP authentication module, you must configure the LTM system to access data on a remote LDAP server. To do this, you must create:

- An LDAP configuration object
- An LDAP profile

After you create these objects, you must then:

- Assign the LDAP profile to a virtual server.
- Assign an LDAP iRule to the virtual server.

Creating an LDAP configuration object

When you create an LDAP configuration object, you configure a variety of settings. Table 8.1 shows the settings and values that you configure for an LDAP configuration object. Note that this table groups the settings into the same categories that you see on the New Authentication Configuration screen. For the detailed procedure on how to create this object, see *To create an LDAP configuration object*, on page 8-6.

Setting	Description	Default Value
General Properties		
Name	Specifies a unique name for the configuration object. This setting is required.	No default value
Type	Specifies the type of authentication module you want to implement. You must set this value to **LDAP**.	No default value
Configuration		
Remote LDAP Tree	Specifies the search base distinguished name.	No default value
Hosts	Lists the addresses of the LDAP servers that the LTM system uses to obtain authentication data.	No default value
Service Port	Specifies the port number for the LDAP service.	**389** (non-SSL) **636** (SSL-enabled)
LDAP Version	Specifies the version number of the LDAP application.	**3**

Table 8.1 *Settings of an LDAP configuration object*

Setting	Description	Default Value
Bind DN	Specifies 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 setting is only required when a site does not allow anonymous searches. If the remote server is a Microsoft® Windows Active Directory server, the distinguished name must be in the form of an email address.	No default value
Bind Password	Specifies the password for the *search* account created on the LDAP server. This setting is required if you use a **bind** DN.	No default value
Confirm Bind Password	Confirms the password for the **bind** distinguished name. This setting is optional.	No default value
Search Time Limit	Specifies a time limit for a search.	**30**
Bind Time Limit	Specifies a Bind time limit.	**30**
Filter	Specifies a filter. This setting is used for authorizing client traffic.	No default value
Login Attribute	Specifies a login attribute. Normally, the value for this setting is **uid**; however, if the server is a Microsoft Windows Active Directory server, the value must be the account name **SAMACCOUNTNAME** (case-insensitive).	No default value
Group DN	Specifies the group distinguished name. This setting is used for authorizing client traffic.	No default value
Group Member Attribute	Specifies an group member attribute. This setting is used for authorizing client traffic.	No default value
SSL	Allowed values are: **Enabled**, **Disabled**, and **Start TLS**. Note that when enabled, the LTM system changes the service port number from **389** to **636**.	**Disabled**
Check SSL Peer	When enabled, checks an SSL peer.	**Disabled**
SSL CA Certificate	Specifies the name of an SSL CA certificate. Allowed values are: **None**, **Default**, and **Specify Full Path.**	**None**
SSL Client Key	Specifies the name of an SSL client key. Allowed values are: **None**, **Default**, and **Specify Full Path.**	**None**
SSL Client Certificate	Specifies the name of an SSL client certificate. Allowed values are: **None**, **Default**, and **Specify Full Path.**	**None**
SSL Ciphers	Specifies SSL ciphers.	No default value
Ignore Unknown User	When enabled, causes the LTM system to ignore unknown users.	**Disabled**
Warning Logging	Enables or disables warning messages.	**Enabled**
Debug Logging	Enables or disables SYSLOG debugging information at LOG_DEBUG level. Not recommended for normal use.	**Disabled**

Table 8.1 *Settings of an LDAP configuration object*

To create an LDAP configuration object

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Configurations.
4. In the upper right corner of the screen, click **Create**.
   This displays the New Configuration screen.
5. For the **Name** setting, type a unique name for the configuration object, such as **my_ldap_config**.
6. For the **Type** setting, select **LDAP**.
   The screen expands to show several settings.
7. Modify or retain values for all settings shown.
   (To configure advanced settings, locate the **Configuration** heading and select **Advanced**.)
8. Click **Finished**.

Creating an LDAP profile

Once you have created an LDAP configuration object, you must create or configure an LDAP profile. You do this by modifying the default **ldap** profile or by creating a custom profile that inherits the default profile settings. An important function of the authentication profile is to reference an existing configuration object that you have created.

In most cases, the default profile should suit your needs. However, even if you use the default profile, you must still modify it to specify the corresponding configuration object that you created.

If you choose to create a custom profile, you must specify a parent profile (either a custom profile or the default profile) that contains the values that you want the new profile to inherit.

When you create an LDAP profile, you configure some settings. Table 8.2 shows the settings and values that make up an LDAP profile. For the detailed procedure on creating an LDAP profile, see *To modify the default LDAP profile*, on page 8-7, or *To create a custom LDAP profile*, on page 8-7.

Setting	Description	Default Value
Name	Specifies a unique name for the profile. This setting is required.	No default value
Type	Specifies the type of authentication module you want to implement. You must set this value to **LDAP**.	No default value
Parent Profile	Specifies the profile from which you want to inherit values.	**ldap**

Table 8.2 *Settings of an LDAP profile*

Setting	Description	Default Value
Mode	Specifies whether the profile is enabled or disabled. Possible settings are **Auto**, **Enabled**, and **Disabled**.	**Enabled**
Configuration	Specifies an existing LDAP configuration object. This setting is required.	No default value
Rule	Specifies the name of an existing authentication iRule. If you do not specify an iRule, the LTM system uses the corresponding default iRule.	**auth_ldap**
Idle Timeout	Specifies the duration in seconds that an authentication or authorization request is idle before timing out. You can use the default value, specify a different value, or select **Indefinite**.	**300**

Table 8.2 *Settings of an LDAP profile*

To modify the default LDAP profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Profiles.
   This displays the list of default authentication profiles.
4. In the Name column, click **ldap**.
5. For the **Mode** setting, select **Enabled** or **Auto**.
6. For the **Configuration** setting, select a configuration object from the list. Note that **None** is not an allowed value.
7. For the **Rule** setting, specify an authentication iRule:
   - If you want to use the default iRule **auth_ldap**, leave the setting as is.
   - If you do not want to use the default iRule **auth_ldap**, select the name of an existing iRule that you have created.
8. Click **Finished**.

To create a custom LDAP profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Profiles.
4. In the upper right corner of the screen, click **Create**.
   This displays the New Profile screen.
5. For the **Name** setting, type a unique name for the profile, such as **my_ldap_profile**.

6. For the **Type** setting, select **LDAP**.
   The screen expands to show additional settings.
7. For the **Parent Profile** setting:
   - If you want to use the default profile **ldap** for the parent profile, leave the setting as is.
   - If you want to use a custom profile for the parent profile, select a custom profile name from the list.
8. For the **Mode** setting, select **Enabled** or **Auto**.
9. For the **Configuration** setting, select a configuration object from the list.
10. For the **Rule** setting, specify an authentication iRule:
    - If you want to use the default iRule **auth_ldap**, leave the setting as is.
    - If you do not want to use the default iRule **auth_ldap**, select the name of an existing iRule that you have created.
11. Click **Finished**.

After you have created an LDAP configuration object and an LDAP profile, you must do the following:

- Assign the profile to the virtual server by configuring the virtual server's **Authentication Profile** setting.
- Assign the default **auth_ldap** iRule to the virtual server by configuring the virtual server's **Rules** setting.

For information on how to configure virtual server settings, see Chapter 2, *Configuring Virtual Servers*.

Implementing a RADIUS authentication module

A RADIUS authentication module is a mechanism for authenticating client connections passing through an LTM system. You use this module when your authentication data is stored on a remote RADIUS server. In this case, client credentials are based on basic HTTP authentication (that is, user name and password).

To implement a RADIUS authentication module, you must configure the LTM system to access data on a remote RADIUS server. To do this, you must create:

- One or more high-level RADIUS server objects
- A RADIUS configuration object
- A RADIUS profile

After you create these objects, you must then:

- Assign the RADIUS profile to a virtual server.
- Assign a RADIUS iRule to the virtual server.

Creating a RADIUS server object

When you create a RADIUS server object, you configure some settings. Table 8.3 shows the settings and values that make up a default RADIUS server object. For the detailed procedure on creating a server object, see *To create a RADIUS server object*, on page 8-10.

Setting	Description	Default Value
Name	Specifies a unique name for the RADIUS server object, such as **my_radius_object**. This setting is required.	No default value
Host	Specifies a host name or IP address for the RADIUS server. This setting is required.	No default value
Service Port	Specifies the port for RADIUS authentication traffic.	**1812**
Secret	Sets the secret key used to encrypt and decrypt packets sent or received from the server. This setting is required.	No default value
Confirm Secret	Confirms the secret key supplied for the **Secret** setting. This setting is required.	No default value
Timeout	Specifies a timeout value.	**3**

Table 8.3 *Settings of a RADIUS sever definition*

To create a RADIUS server object

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose RADIUS Servers.
   This displays the RADIUS Server List screen.
4. In the upper right corner of the screen, click **Create**.
5. For the **Name** setting, type a unique name for the RADIUS server object, such as **my_radius_server**.
6. For the **Server** setting, type a host name or IP address for the remote RADIUS server.
7. For the **Secret** and **Confirm Secret** settings, type the RADIUS secret.
8. Retain or modify all other settings.
9. Click **Finished**.
10. For redundant RADIUS servers, repeat these steps to create additional server objects.

Creating a RADIUS configuration object

When you create a RADIUS configuration object, you configure a variety of settings. Table 8.4 shows the settings and values that make up a RADIUS configuration object. Note that this table groups the settings into the same categories that you see on the New Authentication Configuration screen. For the detailed procedure on how to create this configuration object, see *To create a RADIUS configuration object*, on page 8-11.

Setting	Description	Default Value
Name	Specifies a unique name for the configuration object. This setting is required.	No default value
Type	Specifies the type of authentication module you want to implement. You must set this value to **RADIUS**.	No default value
RADIUS Servers	Lists the IP addresses of the RADUIS servers that LTM will use to obtain authentication data. Note that for each server listed, you must create a corresponding RADIUS server definition. A RADIUS server definition specifies the server name, port number, RADIUS secret, and timeout value. For more information, see Table 8.3.	No default value
Client ID	Sends a NAS-Identifier RADIUS attribute with string bar. If you do not specify a value for the **Client ID** setting, the PAM service type is used instead. This feature can be disabled by specifying a blank client ID.	No default value

Table 8.4 *Settings of a RADIUS configuration object*

Setting	Description	Default Value
Debug Logging	Enables SYSLOG debugging information at LOG_DEBUG level. We do not recommend this for normal use.	**Disable**
Accounting Bug	Disables validation of the accounting response vector. This option should only be necessary on older servers.	**Disable**
Retries	Specifies the number of authentication retries that the LTM system allows before authentication fails.	**3**

Table 8.4 *Settings of a RADIUS configuration object*

To create a RADIUS configuration object

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Configurations.
4. In the upper right corner of the screen, click **Create**.
   This displays the New Configuration screen.
5. For the **Name** setting, specify a unique name for the configuration object, such as **my_radius_config**.
6. For the **Type** setting, select **RADIUS**.
   The screen expands to show several settings.
7. Modify or retain values for all settings shown.
   (To configure advanced settings, locate the **Configuration** heading and select **Advanced**.)
8. Click **Finished**.

Creating a RADIUS profile

Once you have created a RADIUS configuration object, you must create or configure a RADIUS profile. You do this by modifying the default **radius** profile or by creating a custom profile that inherits the default profile settings. An important function of the authentication profile is to reference an existing configuration object.

In most cases, the default profile should suit your needs. However, even if you use the default profile, you must still modify it to specify the corresponding configuration object that you created.

If you choose to create a custom profile, you must specify a parent profile (either a custom profile or the default profile) that contains the values that you want the new profile to inherit.

When you create a RADIUS profile, you configure a variety of settings. Table 8.5 shows the settings and values that make up a RADIUS profile. For the detailed procedure on creating a RADIUS profile, see *To modify the default RADIUS profile*, on page 8-12 or *To create a custom RADIUS profile*, on page 8-13.

Setting	Description	Default Value
Name	Specifies a unique name for the profile. This setting is required.	No default value
Type	Specifies the type of authentication module you want to implement. You must set this value to **RADIUS**.	No default value
Parent Profile	Specifies the profile from which you want to inherit values.	**radius**
Mode	Specifies whether the profile is enabled or disabled. Possible settings are **Auto**, **Enabled**, and **Disabled**.	**Enabled**
Configuration	Specifies an existing RADIUS configuration object.	No default value
Rule	Specifies the name of an existing authentication rule. If you do not specify an iRule, the LTM system uses the corresponding default iRule.	**auth_radius**
Idle Timeout	Specifies the duration in seconds that an authentication or authorization request is idle before timing out. You can use the default value, specify a different value, or select **Indefinite**.	**300**

Table 8.5 *Settings of a RADIUS profile*

To modify the default RADIUS profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Profiles.
   This displays the list of default authentication profiles.
4. In the Name column, click **radius**.
5. For the **Mode** setting, select **Enabled** or **Auto**.
6. For the **Configuration** setting, select a configuration object from the list. Note that **None** is not an allowed setting.
7. For the **Rule** setting, specify an authentication iRule:
   - If you want to use the default iRule **auth_radius**, leave the setting as is.
   - If you do not want to use the default iRule **auth_radius**, select the name of an existing iRule that you have created.
8. Click **Finished**.

To create a custom RADIUS profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Profiles.
4. In the upper right corner of the screen, click **Create**.
   This displays the New Profile screen.
5. In the **Name** setting, type a unique name for the RADIUS profile, such as **my_radius_profile**.
6. In the **Type** setting, select **RADIUS**.
   The screen expands to show additional settings
7. For the **Parent Profile** setting:
   - If you want to use the default profile **radius** for the parent profile, leave the setting as is.
   - If you want to use a custom profile for the parent profile, select a custom profile name from the list.
8. In the **Mode** setting, select **Enabled** or **Auto**.
9. In the **Configuration** setting, select a configuration object from the list.
10. For the **Rule** setting, specify an authentication iRule:
    - If you want to use the default iRule **auth_radius**, leave the setting as is.
    - If you do not want to use the default iRule **auth_radius**, select the name of an existing iRule that you have created.
11. Click **Finished**.

After you have created a RADIUS server object, a RADIUS configuration object, and a RADIUS profile, you must do the following:

- Assign the profile to the virtual server by configuring the virtual server's **Authentication Profile** setting.
- Assign the default **auth_radius** iRule to the virtual server by configuring the virtual server's **Rules** setting.

For information on how to configure virtual server settings, see Chapter 2, *Configuring Virtual Servers*.

Implementing a TACACS+ authentication module

A TACACS+ authentication module is a mechanism for authenticating client connections passing through an LTM system. You use this module when your authentication data is stored on a remote TACACS+ server. In this case, client credentials are based on basic HTTP authentication (that is, user name and password).

To implement the TACACS+ authentication module, you must configure the LTM system to access data on a remote TACACS+ server. To do this, you must create:

- A TACACS+ configuration object
- A TACACS+ profile

After you create these objects, you must then:

- Assign the TACACS+ profile to a virtual server.
- Assign a TACACS+ iRule to the virtual server.

Creating a TACACS+ configuration object

When you create a TACACS+ configuration object, you configure a variety of settings. Table 8.6 shows the settings and values that make up a TACACS+ configuration object. Note that this table groups the settings into the same categories that you see on the New Authentication Configuration screen. For the detailed procedure on how to create this object, see *To create a TACACS+ configuration object*, on page 8-15.

Setting	Description	Default Value
Name	Specifies a unique name for the configuration object. This setting is required.	No default value
Type	Specifies the type of authentication module you want to implement. You must set this value to **TACACS+**.	No default value
Servers	Specifies a host name or IP address for the TACACS++ server. This setting is required.	No default value
Secret	Sets the secret key used to encrypt and decrypt packets sent or received from the server. This setting is required.	No default value
Confirm Secret	Confirms the secret key supplied for the **Secret** setting. This setting is required.	No default value
Encryption	Enables or disables encryption of TACACS+ packets. Recommended for normal use.	**Enabled**
Service Name	Specifies the TACACS+ server's listening device, such as **ppp**.	No default value
Protocol Name	Specifies the TACACS+ server's listening device, such as **lcp**.	No default value

Table 8.6 *Settings of a* TACACS+ *configuration object*

Setting	Description	Default Value
Authentication	Specifies authentication options. Possible values are **Authenticate to first server** and **Authenticate to each server until success**.	**Authenticate to first server**
Accounting Information	If multiple TACACS+ servers are defined and PAM session accounting is enabled, sends accounting start and stop packets to the first available server or to all servers. Possible values are **Send to first available server** and **Send to all servers**.	**Send to first available server**
Debug Logging	Enables SYSLOG debugging information at LOG_DEBUG level. Not recommended for normal use.	**Disable**

Table 8.6 *Settings of a* TACACS+ *configuration object*

To create a TACACS+ configuration object

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Configurations.
4. In the upper right corner of the screen, click **Create**.
   This displays the New Configuration screen.
5. For the **Name** setting, specify a unique name for the configuration object, such as **my_tacacs_config**.
6. For the **Type** setting, select **TACACS+**.
   The screen expands to show several settings.
7. Modify or retain values for all settings shown.
   (To configure advanced settings, locate the **Configuration** heading and select **Advanced**.)
8. Click **Finished**.

Creating a TACACS+ profile

Once you have created a TACACS+ configuration object, you must create or configure a TACACS+ profile. You do this by modifying the default **tacacs+** profile or by creating a custom profile that inherits the default profile settings. An important function of the authentication profile is to reference an existing configuration object.

In most cases, the default profile should suit your needs. However, even if you use the default profile, you must still modify it to specify the corresponding configuration object that you created.

If you choose to create a custom profile, you must specify a parent profile (either a custom profile or the default profile) that contains the values that you want the new profile to inherit.

When you create a TACACS+ profile, you configure a variety of settings. Table 8.7 shows the settings and values that make up a TACACS+ profile. For the detailed procedure on creating a TACACS+ profile, see *To modify the default TACACS+ profile*, on page 8-16 or *To create a custom TACACS+ profile*, on page 8-17.

Setting	Description	Default Value
Name	Specifies a unique name for the profile. This setting is required.	No default value
Type	Specifies the type of authentication module you want to implement. You must set this value to **TACACS+**.	No default value
Parent Profile	Specifies the profile from which you want to inherit values.	**tacacs+**
Mode	Specifies whether the profile is enabled or disabled. Possible settings are **Auto**, **Enabled**, and **Disabled**.	**Enabled**
Configuration	Specifies an existing TACACS+ configuration object.	No default value
Rule	Specifies the name of an existing authentication rule. If you do not specify an iRule, the LTM system uses the corresponding default iRule.	**auth_tacacs+**
Idle Timeout	Specifies the duration in seconds that an authentication or authorization request is idle before timing out. You can use the default value, specify a different value, or select **Indefinite**.	**300**

Table 8.7 *Settings of a TACACS+ profile*

To modify the default TACACS+ profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Profiles.
   This displays the list of default authentication profiles.
4. In the Name column, click **tacacs+**.
5. For the **Mode** setting, select **Enabled** or **Auto**.
6. For the **Configuration** setting, select a configuration object from the list. Note that **None** is not an allowed setting.
7. For the **Rule** setting, specify an authentication iRule:
   - If you want to use the default iRule **auth_tacacs**, leave the setting as is.
   - If you do not want to use the default iRule **auth_tacacs**, select the name of an existing iRule that you have created.
8. Click **Finished**.

To create a custom TACACS+ profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Profiles.
4. In the upper right corner of the screen, click **Create**.
   This displays the New Profile screen.
5. For the **Name** setting, specify a unique name for the profile.
6. For the **Type** setting, select **TACACS+**.
7. For the **Parent Profile** setting:
   - If you want to use the default profile **tacacs** for the parent profile, leave the setting as is.
   - If you want to use a custom profile for the parent profile, select a custom profile name from the list.
8. For the **Mode** setting, select **Enabled** or **Auto**.
9. For the **Configuration** setting, select a configuration object from the list.
10. For the **Rule** setting, specify an authentication iRule:
    - If you want to use the default iRule **auth_tacacs**, leave the setting as is.
    - If you do not want to use the default iRule **auth_tacacs**, select the name of an existing iRule that you have created.
11. Click **Finished**.

After you have created a TACACS+ configuration object and a TACACS+ profile, you must do the following:

- Assign the profile to the virtual server by configuring the virtual server's **Authentication Profile** setting.
- Assign the default **auth_tacacs** iRule to the virtual server by configuring the virtual server's **Rules** setting.

For information on how to configure virtual server settings, see Chapter 2, *Configuring Virtual Servers*.

Implementing an SSL client certificate LDAP authentication module

An SSL client certificate LDAP authentication module is a mechanism for authenticating or authorizing client connections passing through an LTM system. In this case, client credentials are based on SSL certificate credentials instead of user name and password.

LDAP client authorization is based not only on SSL certificates but also on user groups and roles that you define.

Important

*Prior to implementing an SSL client certificate LDAP authentication module, you must configure and implement a **Client SSL** profile. In the profile, the value of the **Mode** configuration setting must be set to **Auto** or **Enabled**.*

Understanding SSL client certificate authorization

With SSL client certificate LDAP authorization, the LTM system can authorize clients based on signed client certificates issued by trusted CAs. Then, to further enhance the ability of the system to authorize client requests, you can also specify groups and roles. Basing authorization on certificates as well as groups and roles provides the flexibility you need to control client access to system resources.

Before you can implement an SSL client certificate LDAP module, you must understand the two different types of credentials that the LTM system uses to authorize application traffic using data on a remote LDAP server. These two types of credentials are:

- SSL certificates
- Groups and roles

Using SSL certificates for LDAP authorization

During the process of authorizing a client, the LTM system must search the LDAP database. When using certificate-based authorization, the system can search the LDAP database in three ways:

- **User**
  Many LDAP server environments already incorporate certificates into the user information stored in the LDAP database. One way of configuring authorization in LDAP server environments is to configure the system 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.

- **Certificate Map**
  If you create an object and class that map certificates to users in the LDAP database, you can then configure the system to search for a certificate in the map, and retrieve a user from that map. The system then checks to ensure that the user in the LDAP database is a valid user.
- **Certificate**
  If certificates are not stored in the LDAP database, you can configure the system to extract a user name from the certificate presented as part of the incoming client request. The system 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 8.8.

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 8.8 *Search results and corresponding authorization status*

Using groups and roles for LDAP 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 LTM system 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 system will search for groups in the LDAP database. Also, you must specify setting values for a group name and a member name. Once you have completed these tasks, the system 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 a setting directly associated with a user. Any role-based authorization that the LTM system performs 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 LTM system searches through the roles assigned to the user and attempts to match that role to a valid role defined by the administrator.

To implement SSL client certificate LDAP authorization, you must configure the LTM system to access data on a remote LDAP server. To do this, you must create:

- An SSL client certificate LDAP configuration object
- An SSL client certificate LDAP profile

After you create these objects, you must then:

- Assign the SSL client certificate LDAP profile to a virtual server.
- Assign a SSL client certificate LDAP iRule to the virtual server.

Creating an SSL client certificate LDAP configuration object

The SSL client certificate LDAP configuration object consists of a set of data and instructions that the corresponding external LDAP server needs when servicing an authorization request from the LTM system.

When you create an SSL client certificate LDAP configuration object, you configure a variety of settings. Table 8.9 lists and describes the settings that you can specify in this configuration object. Note that this table groups the settings into the same categories that you see on the New Authentication Configuration screen. For the detailed procedure on how to configure this object, see *To create an SSL client certificate LDAP configuration object*, on page 8-22.

Setting	Description	Default value
Name	Specifies a unique name for the configuration object. This setting is required.	No default value
Type	Specifies the type of authentication module you want to implement. You must set this value to **SSL Client Certificate LDAP**.	No default value
Hosts	Lists the IP addresses, including port numbers, of the LDAP servers that the LTM system will use to obtain authorization data.	No default value
Search Type	Specifies the certificate-based authorization method that the system uses when searching the LDAP database (described in *Using SSL certificates for LDAP authorization*, on page 8-18). Possible values are **User**, **Certificate Map**, and **Certificate**.	**User**
User Base DN	Specifies the search base for the subtree used by the **User** and **Certificate** search types. A typical search base is: **ou=people,dc=company,dc=com**. This setting is required. For more information, see the **Search Type** setting.	No default value

Table 8.9 *Settings of an SSL client certificate LDAP configuration object*

Setting	Description	Default value
User Key	Specifies the name of the attribute in the LDAP database that specifies a user ID. Used by the **User** and **Certificate** search type. A typical example of a user key value is **uid**. This setting is required. For more information, see the **Search Type** setting.	No default value
Object Class	Specifies a user authentication method. This setting only appears when you set the search type to **Certificate**.	**StrongAuthenticationUser**
Certificate Map Base DN	Specifies the search base for the subtree used by the **Certificate Map** search types. A typical search base is: **ou=people,dc=company,dc=com**. This setting is required. For more information, see the **Search Type** setting.	No default value
Certificate Map Key	Specifies the name of the attribute in the LDAP database that specifies a user ID. Used by the **Certificate Map** search type. A typical example of a user key value is **uid**. This setting is required. For more information, see the **Search Type** setting.	No default value
Use Serial Certificate Map	Causes the system to use the serial number of the client certificate instead of the subject, when you the select **Certificate Map** search type.	**Disabled**
Cache Size	Specifies the maximum size allowed for the SSL session cache. Setting this value to **0** disallows SSL session caching.	**20000**
Cache Timeout	Specifies the number of usable lifetime seconds of negotiable SSL session IDs. If this time has expired, a client must negotiate a new session.	**300**
Secure	Instructs the LTM system to use secure communication (that is, SSL/TLS) between the system and the LDAP server.	**Disabled**
Admin DN	Specifies 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 setting is required only when a site does not allow anonymous searches.	No default value
Admin Password	Specifies the password for the *search* account created on the LDAP server.	No default value
Confirm Admin Password	Confirms the password specified for the *search* account created on the LDAP server.	No default value

Table 8.9 *Settings of an SSL client certificate LDAP configuration object*

Setting	Description	Default value
Group Base DN	Specifies 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**.	No default value
Group Key	Indicates the name of the attribute in the LDAP database that specifies the group name in the group subtree.	No default value
Group Member Key	Indicates the name of the attribute in the LDAP database that specifies members (DNs) of a group.	No default value
Valid Groups	Specifies a list of groups to which a user must belong in order to be authorized. This option may be specified more than once. To be authorized, a client need only be a member of a single group specified in the string.	No default value
Role Key	Indicates 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).	No default value
Valid Roles	Specifies 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.This option may be specified more than once. To be authorized, a client need only be a member of a single role specified in the string.	No default value

Table 8.9 *Settings of an SSL client certificate LDAP configuration object*

To create an SSL client certificate LDAP configuration object

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Configurations.
4. In the upper right corner of the screen, click **Create**.
   This displays the New Configuration screen.
5. For the **Name** setting, specify a unique name for the configuration object.
6. For the **Type** setting, select **SSL Client Certificate LDAP**.
7. Modify or retain values for all other settings. (To configure advanced settings, locate the **Configuration** heading and select **Advanced**.)
8. Click **Finished**.

Creating an SSL client certificate LDAP authorization profile

Once you have created an SSL client certificate LDAP configuration object, you must create or configure a corresponding profile. You do this by modifying the default **ssl_cc_ldap** profile or by creating a custom profile that inherits the default profile settings. An important function of the authentication profile is to reference an existing configuration object.

In most cases, the default profile should suit your needs. However, even if you use the default profile, you must still modify it to specify the corresponding configuration object that you created.

If you choose to create a custom profile, you must specify a parent profile (either a custom profile or the default profile) that contains the values that you want the new profile to inherit.

When you create an SSL client certificate LDAP profile, you configure a variety of settings. Table 8.10 shows the settings and values that make up an SSL client certificate LDAP profile. For the detailed procedure on creating this type of profile, see *To modify the default SSL client certificate LDAP profile*, on page 8-23, or *To create a custom SSL client certificate LDAP profile*, on page 8-24.

Setting	Description	Default Value
Name	Specifies a unique name for the profile. This setting is required.	No default value
Type	Specifies the type of authentication module you want to implement. You must set this value to **LDAP (SSL Client Certificate)**.	No default value
Parent Profile	Specifies the profile from which you want to inherit values.	**ssl_cc_ldap**
Mode	Specifies whether the profile is enabled or disabled. Possible settings are **Auto**, **Enabled**, and **Disabled**.	**Enabled**
Configuration	Specifies an existing SSL client certificate LDAP configuration object.	No default value
Rule	Specifies the name of an existing authentication iRule. If you do not specify an iRule, the LTM system uses the corresponding default iRule.	**auth_ssl_cc_ldap**
Idle Timeout	Specifies the duration in seconds that an authorization request is idle before timing out. You can use the default value, specify a different value, or select **Indefinite**.	**300**

Table 8.10 *Settings of an SSL client certificate LDAP profile*

To modify the default SSL client certificate LDAP profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.

3. From the Authentication menu, choose Profiles.
   This displays the list of default authentication profiles.
4. In the Name column, click **ssl_cc_ldap**.
5. For the **Mode** setting, select **Enabled** or **Auto**.
6. For the **Configuration** setting, select a configuration object from the list. Note that **None** is not an allowed setting.
7. For the **Rule** setting, specify an authentication iRule:
   - If you want to use the default iRule **auth_ssl_cc_ldap**, leave the setting as is.
   - If you do not want to use the default iRule **auth_ssl_cc_ldap**, select the name of an existing iRule that you have created.
8. Click **Finished**.

To create a custom SSL client certificate LDAP profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Profiles.
4. In the upper right corner of the screen, click **Create**.
   This displays the New Profile screen.
5. For the **Name** setting, specify a unique name for the profile.
6. For the **Type** setting, select **SSL Client Certificate LDAP**.
7. For the **Parent Profile** setting:
   - If you want to use the default profile **ssl_cc_ldap** for the parent profile, leave the setting as is.
   - If you want to use a custom profile for the parent profile, select a custom profile name from the list.
8. For the **Mode** setting, click the Custom box on the right side of the screen, and select **Enabled** or **Auto**.
9. For the **Configuration** setting, click the Custom box on the right side of the screen, and select a configuration object from the list.
10. For the **Rule** setting, Click the Custom box on the right side of the screen, and specify an authentication iRule:
    - If you want to use the default iRule **auth_ssl_cc_ldap**, leave the setting as is.
    - If you do not want to use the default iRule **auth_ssl_cc_ldap**, select the name of an existing iRule that you have created.
11. Click **Finished**.

After you have created an SSL Client Certificate LDAP configuration object and an SSL Client Certificate LDAP profile, you must do the following:

- Assign the profile to the virtual server by configuring the virtual server's **Authentication Profile** setting.
- Assign the default **auth_ssl_cc_ldap** iRule to the virtual server by configuring the virtual server's **Rules** setting.

Implementing an SSL OCSP authentication module

An SSL OCSP authentication module is a mechanism for authenticating client connections passing through an LTM system. More specifically, an SSL OCSP authentication module checks the revocation status of an SSL certificate, as part of authenticating that certificate.

Online Certificate Status Protocol (OCSP) is a third-party software application and industry-standard protocol that offers an alternative to a certificate revocation list (CRL) when using public-key technology. A ***CRL*** is a list of revoked client certificates, which a server system can check during the process of verifying a client certificate.

You implement an SSL OCSP authentication module when you want to use OCSP instead of a CRL as the mechanism for checking the revocation status of SSL certificates.

The LTM system supports both CRLs and the OCSP protocol. If you want to use CRLs instead of OCSP, you configure an SSL profile. For more information on CRLs, see Chapter 7, *Managing SSL Traffic*.

For more information on OCSP, see RFC 2560 at URL ***http://www.ietf.org***.

Understanding OCSP

Using OCSP to check on the revocation status of client certificates offers distinct advantages over the use of a CRL. The following sections describe the differences between CRLs and OCSP, as well as the way that OCSP operates.

The limitations of CRLs

When presented with a client certificate, the LTM system sometimes needs to assess the revocation state of that certificate before accepting the certificate and forwarding the connection to a target server. The standard method of assessing revocation status is a CRL, which is stored in a separate CRL file on each machine in your configuration. Although CRLs are considered to be a standard way of checking revocation status of SSL certificates, a CRL is updated only at fixed intervals, thus presenting a risk that the information in the CRL is outdated at the time that the status check occurs.

Also, having to store a separate CRL file on each machine presents other limitations:

- All CRL files must be kept in sync.
- Having a separate CRL file on each machine poses a security risk.
- Multiple CRL files cannot be administered from a central location.

The benefits of OCSP

OCSP ensures that the LTM system always obtains real-time revocation status during the certificate verification process.

OCSP is based on a client/server model, where a client system requests revocation status of a certificate, and a server system sends the response. Thus, when you implement the SSL OCSP authentication module, the LTM system acts as the OCSP client, and an external system, known as an OCSP responder, acts as the OCSP server. An ***OCSP responder*** is therefore an external server that sends certificate revocation status, upon request, to the LTM system.

How does OCSP work?

In general, after receiving an SSL certificate from a client application, the LTM system (acting as an OCSP client) requests certificate revocation status from an OCSP responder, and then blocks the connection until it receives status from that responder. If the status from the responder shows that the certificate has been revoked, the LTM system rejects the certificate and denies the connection. If the status from the responder shows that the certificate is still valid, the LTM continues with its normal certificate verification process to authenticate the client application.

More specifically, when an application client sends a certificate for authentication, the LTM system follows this process:

- First, the LTM system checks that the signer of the certificate is listed in the trusted CAs file.
- If the certificate is listed, the LTM system then checks to see if the certificate has been revoked. Without OCSP, if the CRL option is configured on the LTM system, the LTM system checks revocation status by reading the certificate revocation list (CRL). With OCSP, however, the LTM system bypasses the CRL and prepares to send a revocation status request to the appropriate OCSP responder.
- The LTM system then chooses the OCSP responder by checking the CA specified in the **Issuer** field of the original client certificate.
- Next, the LTM system attempts to match that CA with a CA listed in an SSL OCSP profile.
- If a match exists, the LTM system checks the target URL within the client certificate's **AuthorityInfoAccess** (**AIA**) field (if the field exists), and uses that URL to send the request for certificate revocation status to the OCSP responder.
- If the **Ignore AIA** parameter is enabled within the SSL OCSP profile, then the LTM system instead uses the URL specified in the **url** parameter of the matching SSL OCSP profile to send the request for certificate revocation status.
- If no match exists, the LTM system checks the **calist** setting of another SSL OCSP profile defined on the system. If all SSL OCSP profiles are checked and no match is found, the certificate verification fails, and the LTM system denies the original client request.

- Once the LTM system has received certificate revocation status from a responder, the LTM system, when configured to do so, inserts a certificate status header into the original client request. The name of the certificate status header is **SSLClientCertificateStatus**. For more information on this header, see *Prerequisite LTM profiles*, following.

To implement the SSL OCSP authentication module, you must configure the LTM system to access data on a remote OCSP server. To do this, you must create:

- An OCSP configuration object
- An OCSP profile

After you create these objects, you must then:

- Assign the OCSP profile to a virtual server.
- Assign an OCSP iRule to the virtual server.

A single SSL OCSP profile can target a specific responder, or multiple SSL OCSP profiles can target the same responder. Each responder itself is associated with a certificate authority (CA), and multiple responders can be associated with the same CA.

◆ Note

The LTM system allows you to enable both the CRL and the OCSP options. Most users need to enable either one or the other, but not both. However, in the rare case that you want to enable both options, be aware that both the search of the CRL file, and the connection to the responder must be successful. Otherwise, the LTM system cannot obtain status.

Prerequisite LTM profiles

Configuring an SSL OCSP authentication module not only requires the creation of an OCSP responder object and SSL OCSP profile, but also the creation of two other profiles: HTTP and Client SSL.

When you create a Client SSL profile, we recommend that you configure the profile to request, but not require, certificates. This optimizes the LTM system's use of the **SSLClientCertificateStatus** header. Note that the LTM system only inserts this header when previously configured to insert headers into client requests (either through an HTTP profile or through an iRule).

The following sections describe how to create an OCSP responder object, an SSL OCSP configuration object, and an SSL OCSP profile.

Creating an OCSP responder object

An OCSP responder object is an object that you create that includes a URL for an external OCSP responder. You must create a separate OCSP responder object for each external OCSP responder.

When you subsequently create an OCSP configuration object, the configuration object contains a reference to any OCSP responder objects that you have created.

When you create an OCSP responder object, you configure some settings. Table 8.11 shows the settings and values that make up an SSL OCSP responder object. For the detailed procedure on how to create this object, see *To create an OCSP responder object*, on page 8-31.

Setting	Description	Default Value
Name	Specifies a unique name for the configuration object. This setting is required.	No default value
URL	Specifies the URL used to contact the OCSP service on the responder.	No default value
Certificate Authority File	Specifies the name of the file containing trusted CA certificates used to verify the signature on the OCSP response.	No default value
Certificate Authority Path	Specifies the name of the path containing trusted CA certificates used to verify the signature on the OCSP response.	No default value
Verify Other	Specifies the name of the file used to search for an OCSP response signing certificate when the certificate has been omitted from the response.	No default value
Trust Other	Instructs the LTM system to trust the certificates specified with the **Verify Other** setting.	**Disable**
VA File	Specifies the name of the file containing explicitly-trusted responder certificates. This parameter is needed in the event that the responder is not covered by the certificates already loaded into the responder's CA store.	No default value
Signer	Specifies the certificate used to sign an OCSP request. Special meanings: If the certificate is specified but the key is not specified, then the private key is read from the same file as the certificate. If neither the certificate nor the key is specified, then the request is not signed. If the certificate is not specified and the key is specified, then the configuration is considered to be invalid.	No default value
Signing Key	Specifies a key used to sign an OCSP request.	No default value
Sign Other	Lists additional certificates to add to an OCSP request.	No default value

Table 8.11 *Settings of an OCSP responder*

Setting	Description	Default Value
Sign Digest	Specifies the algorithm for signing the request, using the signing certificate and key. Special meanings: This parameter has no meaning if request signing is not in effect (that is, both the **request signing certificate** and **request signing key** parameters are empty). This parameter is required only when request signing is in effect.	**Sha1**
Validity Period	Specifies the number of seconds that the LTM system should use to specify an acceptable error range. This setting is used when the OCSP responder clock and a client clock are not synchronized, which could cause a certificate status check to fail. This value must be a positive number. This parameter is required.	**300**
Status Age	Specifies a time in seconds to compare to the **notBefore** field of a status response. Used when the status response does not include the **notAfter** field.	**0**
Ignore AIA	Instructs the LTM system to ignore the URL contained in the certificate's **AIA** fields, and to always use the URL that the responder specifies instead.	Disabled
Trust Other	Specifies that the certificates should be explicitly trusted and no other checks should be performed on them.	Disabled
Allow Certificates	Allows the addition of certificates to an OCSP request.	Enabled
Verify	Causes the LTM system to verify an OCSP response signature or the nonce values. Used for debugging purposes only.	Enabled
Intern	Causes the LTM system to ignore certificates contained in an OCSP response when searching for the signer's certificate. To use this setting, the signer's certificate must be specified with either the **Verify Other** or **VA File** setting.	Enabled
Verify Signature	Causes the LTM system to check the signature on the OCSP response. Used for testing purposes only.	Enabled
Verify Certificate	Causes the LTM system to verify the certificate in the OCSP response.	Enabled
Certificate Chain	Causes the LTM system to construct a chain from certificates in the OCSP response.	Enabled
Check Certificates	Causes the LTM system to make additional checks to see if the signer's certificate is authorized to provide the necessary status information. Used for testing purposes only	Enabled
Explicit OCSP	Causes the LTM system to explicitly trust that the OCSP response signer's certificate is authorized for OCSP response signing. If the signer's certificate does not contain the **OCSP signing** extension, specification of this setting causes a response to be untrusted.	Enabled

Table 8.11 *Settings of an OCSP responder*

To create an OCSP responder object

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles.**
3. From the Authentication menu, choose OCSP Responders.
4. In the upper right corner of the screen, click **Create**.
5. For the **Name** setting, type a unique name for the responder object, such as **my_ocsp_responder**.
6. For the **URL** setting, type a URL for the external responder.
7. Modify or retain values for all other settings. (To configure advanced settings, locate the **Configuration** heading and select **Advanced**).
8. Click **Finished**.

Creating an SSL OCSP configuration object

The SSL OCSP configuration object is an object that references one or more OCSP responder objects.

When you create an OCSP configuration object, you configure a variety of settings. Table 8.12 lists and describes the settings that you can specify in an SSL OCSP configuration object. For the detailed procedure on how to configure this object, see *To create an SSL OCSP configuration object*, following.

Setting	Description	Default Value
Name	Specifies a unique name for the configuration object. This setting is required.	No default value
Type	Specifies the type of authentication module you want to implement. You must set this value to **SSL OCSP**.	No default value
Responders	Specifies the OCSP responders that you want to use for checking that revocation status of SSL certificates.	No default value

Table 8.12 *Settings of an SSL OCSP configuration object*

To create an SSL OCSP configuration object

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Configurations.
4. In the upper right corner of the screen, click **Create**.
   This displays the New Configuration screen.

5. For the **Name** setting, specify a unique name for the configuration object, such as **my_ocsp_config**.
6. For the **Type** setting, select **SSL OCSP**.
7. For the **Responders** setting, specify all responder objects.
8. Click **Finished**.

Creating an SSL OCSP profile

Once you have created an SSL OCSP configuration object, you must create or configure an SSL OCSP profile. You do this by modifying the default **ssl_ocsp** profile or by creating a custom profile that inherits the default profile settings. An important function of the authentication profile is to reference an existing configuration object.

In most cases, the default profile should suit your needs. However, even if you use the default profile, you must still modify it to specify the corresponding configuration object that you created.

If you choose to create a custom profile, you must specify a parent profile (either a custom profile or the default profile) that contains the values that you want the new profile to inherit.

When you create an OCSP profile, you configure a variety of settings. Table 8.13 shows the settings and values that make up an SSL OCSP profile. For the detailed procedure on creating an OCSP profile, see *To modify the default SSL OCSP profile*, on page 8-33, or *To create a custom SSL OCSP profile*, on page 8-33.

Setting	Description	Default Value
Name	Specifies a unique name for the profile. This setting is required.	No default value
Type	Specifies the type of authentication module you want to implement. You must set this value to **OCSP**.	No default value
Parent Profile	Specifies the profile from which you want to inherit values.	**ssl_ocsp**
Mode	Specifies whether the profile is enabled or disabled. Possible settings are **Auto**, **Enabled**, and **Disabled**.	**Enabled**
Configuration	Specifies the name of an existing OCSP configuration object. This setting is required.	No default value
Rule	Specifies the name of an existing authentication iRule. If you do not specify an iRule, the LTM system uses the corresponding default iRule.	**auth_ssl_ocsp**
Idle Timeout	Specifies the duration in seconds that an authentication request is idle before timing out. You can use the default value, specify a different value, or select **Indefinite**.	**300**

Table 8.13 *Settings of an SSL OCSP profile*

To modify the default SSL OCSP profile

1. On the Main tab, expand **Local Traffic.**
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Profiles.
   This displays the list of default authentication profiles.
4. In the Name column, click **ssl_ocsp**.
5. For the **Mode** setting, select **Enabled** or **Auto**.
6. For the **Configuration** setting, select a configuration object from the list. Note that **None** is not an allowed setting.
7. For the **Rule** setting, specify an authentication iRule:
   - If you want to use the default iRule **auth_ocsp**, leave the setting as is.
   - If you do not want to use the default iRule **auth_ocsp**, select the name of an existing iRule that you have created.
8. Click **Finished**.

To create a custom SSL OCSP profile

1. On the Main tab, expand **Local Traffic**.
2. Click **Profiles**.
   The Profiles screen opens.
3. From the Authentication menu, choose Profiles.
4. In the upper right corner of the screen, click **Create**.
   This displays the New Profile screen.
5. For the **Name** setting, specify a unique name for the profile.
6. For the **Type** setting, select **SSL OCSP**.
7. For the **Parent Profile** setting:
   - If you want to use the default profile **ssl_ocsp** for the parent profile, leave the setting as is.
   - If you want to use a custom profile for the parent profile, select a custom profile name from the list.
8. For the **Mode** setting, click the Custom box on the right side of the screen, and select **Enabled** or **Auto**.
9. For the **Configuration** setting, click the Custom box on the right side of the screen, and select a configuration object from the list.

10. For the **Rule** setting, specify an authentication iRule:
    - If you want to use the default iRule **auth_ocsp**, leave the setting as is.
    - If you do not want to use the default iRule **auth_ocsp**, select the name of an existing iRule that you have created.
11. Click **Finished**.

After you have created an SSL OCSP responder object, an SSL OCSP configuration object, and an SSL OCSP profile, you must do the following:

- Assign the profile to the virtual server by configuring the virtual server's **Authentication Profile** setting.
- Assign the default **auth_ssl_ocsp** iRule to the virtual server by configuring the virtual server's **Rules** setting.

For information on how to configure virtual server settings, see Chapter 2, *Configuring Virtual Servers*.

9

Enabling Session Persistence

- Introducing session persistence
- Persistence types and their profiles

Introducing session persistence

Using the BIG-IP® local traffic management (LTM) system, you can configure session persistence. When you configure ***session persistence***, the LTM system tracks and stores session data, such as the specific pool member that serviced a client request. The primary reason for tracking and storing session data is to ensure that client requests are directed to the same pool member throughout the life of a session or during subsequent sessions.

In addition, session persistence can track and store other types of information, such as user preferences or a user name and password.

The LTM system offers several types of session persistence, each one designed to accommodate a specific type of storage requirement for session data. The type of persistence that you implement depends on where and how you want to store client-specific information, such as items in a shopping cart or airline ticket reservations.

For example, you might store airline ticket reservation information in a back-end database that all servers can access, or on the specific server to which the client originally connected, or in a cookie on the client's machine. When you enable persistence, returning clients can bypass load balancing and instead connect to the server to which they last connected in order to access their saved information.

The LTM system keeps session data for a period of time that you specify.

The primary tool for configuring session persistence is to configure a persistence profile and assign it to a virtual server. If you want to enable persistence for specific types of traffic only, as opposed to all traffic passing through the virtual server, you can write an iRule.

Configuring a persistence profile

A ***persistence profile*** is a pre-configured object that automatically enables persistence when you assign the profile to a virtual server. By using a persistence profile, you avoid having to write a program to implement a type of persistence.

Each type of persistence that the LTM system offers includes a corresponding default persistence profile. These persistence profiles each contain settings and setting values that define the behavior of the LTM system for that type of persistence. You can either use the default profile or create a custom profile based on the default.

For more information, see the following chapters of this guide:

- To configure persistence profiles, see *Persistence types and their profiles*, on page 9-3.
- To understand profiles in general, see Chapter 5, *Understanding Profiles*.

Enabling session persistence through iRules

Instead of configuring a persistence profile, which enables a persistence type for all sessions passing through the virtual server, you can write an iRule, which enables a persistence type for particular requests (for example, for HTTP traffic that includes a certain cookie version only).

You can also use an iRule to enable persistence for SSL-terminated requests, that is, requests that the LTM system terminates by performing decryption and re-encryption and by handling SSL certificate authentication. In this type of iRule, you can use an HTTP header insertion iRule command to insert an SSL session ID as a header into an HTTP request.

The remainder of this chapter focuses on enabling persistence using persistence profiles. For information on enabling persistence by writing an iRule, see Chapter 13, *Writing iRules*.

Persistence types and their profiles

You can configure persistence profile settings to set up session persistence on your LTM system. You can configure these settings when you create a profile or after profile creation by modifying the profile's settings. For specific procedures on configuring a profile, see Chapter 5, *Understanding Profiles*.

Types of persistence

The persistence types that you can enable using a persistence profile are:

- **Cookie persistence**
  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.
- **Destination address affinity persistence**
  Also known as sticky persistence, destination address affinity persistence supports TCP and UDP protocols, and directs session requests to the same server based solely on the destination IP address of a packet.
- **Hash persistence**
  Hash persistence allows you to create a persistence hash based on an existing iRule.
- **Microsoft Remote Desktop Protocol persistence**
  Microsoft Remote Desktop Protocol (MSRDP) persistence tracks sessions between clients and servers running the Microsoft® Remote Desktop Protocol (RDP) service.
- **SIP persistence**
  SIP persistence is a type of persistence used for servers that receive Session Initiation Protocol (SIP) messages sent through UDP. SIP is a protocol that enables real-time messaging, voice, data, and video.
- **Source address affinity persistence**
  Also known as simple persistence, source address affinity persistence supports TCP and UDP protocols, and directs session requests to the same server based solely on the source IP address of a packet.
- **SSL persistence**
  SSL persistence is a type of persistence that tracks non-terminated SSL sessions, using the SSL session ID. Even when the client's IP address changes, the LTM system still recognizes the connection as being persistent based on the session ID. Note that the term ***non-terminated SSL sessions*** refers to sessions in which the LTM system does not perform the tasks of SSL certificate authentication and encryption/re-encryption. To enable persistence for terminated SSL sessions, see Chapter 7, *Managing SSL Traffic* and Chapter 13, *Writing iRules*.

- **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 iRules™, defines some sequence of bytes to use as a session identifier.

Understanding criteria for session persistence

Regardless of the type of persistence you are implementing, you can specify the criteria that the LTM system uses to send all requests from a given client to the same pool member. These criteria are based on the virtual server or servers that are hosting the client connection. To specify these criteria, you use the **Match Across Services** and **Match Across Virtual Servers** profile settings. Before configuring a persistence profile, it is helpful to understand these settings.

Specifying the Match Across Services setting

When you enable the **Match Across Services** profile setting, the LTM system attempts to send all persistent connection requests received from the same client, within the persistence time limit, to the same pool member 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.

For example, suppose you configure virtual server mappings where the virtual server **v1:http** has persistence enabled and references the **http_pool** (containing the nodes **n1:http** and **n2:http**) and the virtual server **v1:ssl** has persistence enabled and references the pool **ssl_pool** (containing the nodes **n1:ssl** and **n2:ssl**).

If the client subsequently connects to **v1:ssl**, the LTM system uses the persistence session established with the first connection to determine the pool member that should receive the connection request, rather than the load balancing method. The LTM system should send the third connection request to **n1:ssl**, which uses the same node 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 LTM 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 setting 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.

Specifying the Match Across Virtual Servers setting

You can set the LTM system to maintain persistence for all sessions requested by the same client, regardless of which virtual server hosts each individual connection initiated by the client. When you enable the **Match Across Virtual Servers** setting, the LTM system attempts to send all persistent connection requests received from the same client, within the persistence time limit, to the same pool member. Connection requests from the client that do not use persistence are load balanced according to the currently selected load balancing method.

WARNING

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

Cookie persistence

You can set up the LTM system to use HTTP cookie persistence. ***Cookie persistence*** uses an HTTP cookie stored on a client's computer to allow the client to reconnect to the same pool member previously visited at a web site.

There are four methods of cookie persistence available:

- HTTP Cookie Insert method
- HTTP Cookie Rewrite method
- HTTP Cookie Passive method
- Cookie Hash method

The method you choose to use affects how the cookie is handled by the LTM system when it is returned to the client.

The cookie profile

To implement cookie persistence, you can either use the default **cookie** profile, or create a custom profile. Table 9.1 shows the settings and values that make up the default **cookie** profile.

Setting	Description	Default Value
Name	Specifies a unique name for the profile. This setting is required.	No default value
Persistence Type	Specifies the type of persistence. This setting is required.	**Cookie**
Match Across Services	Specifies that all persistent connections from a client IP address that go to the same virtual IP address also go to the same node.	Disabled

Table 9.1 *Settings of a cookie persistence profile*

Setting	Description	Default Value
Match Across Virtual Servers	Specifies that all persistent connections from the same client IP address go to the same node.	Disabled
Match Across Pools	Specifies that the LTM system can use any pool that contains this persistence entry.	Disabled
Cookie Method	Specifies the type of cookie processing that the LTM system is to use. For more information, see *Specifying the Cookie Method setting*, following.	**HTTP Cookie Insert**
Cookie Name	Specifies the name of the cookie that the LTM system should look for or insert.	This value is autogenerated based on the pool name.
Expiration	Sets the expiration time of the cookie.	**Never Expire**

Table 9.1 *Settings of a cookie persistence profile*

Specifying the Cookie Method setting

You can configure the **Cookie Method** setting in a cookie profile to one of four values.

HTTP Cookie Insert method

If you specify **HTTP Cookie Insert** method within the profile, 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 LTM system. **HTTP Cookie Insert** is the default value for the **Cookie Method** setting.

HTTP Cookie Rewrite method

If you specify **HTTP Cookie Rewrite** method, the LTM system intercepts a **Set-Cookie** header, 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.

Important

*We recommend that you use this method instead of the **HTTP Cookie Passive** method whenever possible.*

The **HTTP Cookie Rewrite** method requires you to set up the cookie created by the server. For the **HTTP Cookie Rewrite** method to succeed, there needs to be a blank cookie coming from the web server for the LTM system to rewrite. With Apache variants, the cookie can be added to every web page header by adding the following entry to 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 can contain only 75 zeros. However, cookies of this size do not allow you to use iRules and persistence together.

HTTP Cookie Passive method

If you specify the HTTP Cookie Passive method, the LTM system does not insert or search for blank **Set-Cookie** headers in the response from the server. This method does not try to set up the cookie. With this method, the server provides the cookie, formatted with the correct server information and timeout.

Important

We recommend that you use the HTTP Cookie Rewrite method instead of the HTTP Cookie Passive method whenever possible.

For the HTTP Cookie Passive method to succeed, there needs to be a cookie coming from the web server with the appropriate server information in the cookie. Using the Configuration utility, you generate a template for the cookie string, with encoding automatically added, and then edit the template to create the actual cookie.

For example, the following string is a generated cookie template with the encoding automatically added, where **[pool name]** is the name of the pool that contains the server, **336260299** is the encoded server address, and **20480** is the encoded port:

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

To create your cookie from this template, type the actual pool name and an 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. Thus, port **80** becomes 80 * 256 + 0 = 20480. Port **1433** (instead of 5 * 256 + 153) becomes 153 * 256 + 5 = 39173.

With Apache variants, the cookie can be added to every web page header by adding the following entry to the **httpd.conf** file:

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

Cookie Hash method

If you specify the Cookie Hash method, the hash method consistently maps a cookie value to a specific node. When the client returns to the site, the LTM system uses the cookie information to return the client to a given node. With this mode, the web server must generate the cookie; the LTM system does not create the cookie automatically as it does when you use the HTTP Cookie Insert method.

Destination address affinity persistence

You can optimize your server array with destination address affinity persistence. ***Destination address affinity persistence***, also known as sticky persistence, directs requests for a certain destination IP address to the same server, regardless of which client made the request.

This type of persistence provides the most benefits when load balancing caching servers. A caching 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 servers, it is necessary to send similar requests to the same server repeatedly. You can use the destination address affinity persistence type to cache a given web page on one server instead of on every server in an array. This saves the other servers from having to duplicate the web page in their cache, wasting memory.

The destination address affinity profile

To implement destination address affinity persistence, you either use the default **dest_addr** profile or create a custom profile. Table 9.2 shows the settings and their values that make up the default **dest_addr** profile.

Setting	Description	Default Value
Name	Specifies a unique name for the profile. This setting is required.	No default value
Persistence Type	Specifies the type of persistence profile. This setting is required.	**Destination Address Affinity**
Match Across Services	Specifies that all persistent connections from a client IP address that go to the same virtual IP address also go to the same node.	Disabled
Match Across Virtual Servers	Specifies that all persistent connections from the same client IP address go to the same node.	Disabled
Match Across Pools	Specifies that the LTM system can use any pool that contains this persistence entry.	Disabled
Mask	Specifies the mask that the LTM system should use before matching with an existing persistence entry.	**255.255.255.255**

Table 9.2 *Settings of a destination address affinity persistence profile*

Hash persistence

Hash persistence allows you to create a persistence hash based on an existing iRule.

The Hash profile

To implement hash persistence, you either use the default **hash** profile or create a custom profile. Table 9.3 shows the settings and their values that make up the default **hash** profile.

Setting	Description	Default Value
Name	Specifies a unique name for the profile. This setting is required.	No default value
Persistence Type	Specifies the type of persistence profile. This setting is required.	**Hash**
Match Across Services	Specifies that all persistent connections from a client IP address that go to the same virtual IP address also go to the same node.	Disabled
Match Across Virtual Servers	Specifies that all persistent connections from the same client IP address go to the same node.	Disabled
Match Across Pools	Specifies that the LTM system can use any pool that contains this persistence entry.	Disabled
iRule	Specifies an iRule to run that determines the persistence entry.	**_sys_auth_ldap**
Timeout	Specifies a timeout value in seconds. Another possible value for this setting is **Indefinite**.	**180**

Table 9.3 *Settings of a hash persistence profile*

Microsoft Remote Desktop Protocol persistence

MSRDP persistence provides an efficient way of load balancing traffic and maintaining persistent sessions between Windows® clients and servers that are running the Microsoft® Remote Desktop Protocol (RDP) service. The recommended scenario for enabling MSRDP persistence feature is to create a load balancing pool that consists of members running Windows.NET Server 2003, Enterprise Edition, or later, where all members belong to a Windows cluster and participate in a Windows session directory.

Benefits of MSRDP persistence

Without MSRDP persistence, Windows 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 MSRDP persistence is enabled, however, a Windows server participating in a session directory always redirects the connection to the same virtual server, instead of to another server directly. The LTM system then sends the connection to the correct Windows server. Also, when MSRDP persistence is enabled on an LTM system and the servers in the pool participate in a session directory, the LTM system load balances an RDP connection according to the way that the user has configured the LTM system for load balancing. Thus, the use of Windows servers and the Session Directory service, combined with the MSRDP persistence feature, provides more sophisticated load balancing and more reliable reconnection when servers become disconnected.

Server Platform issues

By default, the LTM system with MSRDP persistence enabled load balances connections according to the way that the user has configured the LTM system for load balancing, as long as Session Directory is configured on each server in the pool. Because Session Directory is a feature that is only available on Windows.NET Server 2003, Enterprise Edition, or later platforms, each server in the pool must therefore be a Windows.NET Server 2003, Enterprise Edition server if you want to use MSRDP 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 MSRDP persistence but your server platforms are running older versions of Windows (on which Session Directory is not available), you can enable MSRDP persistence in non-default mode. This causes the LTM system to connect a client to the same Windows server by way of the user name that the client provides. Note that enabling MSRDP 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.

Configuring MSRDP persistence with Session Directory

To enable MSRDP persistence in the default mode, you must configure Session Directory on each Windows 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 servers, you must configure the LTM system, by performing tasks such as creating a load-balancing pool and designating your Windows servers as members of that pool.

The following two sections describe LTM configuration tasks that are required to enable MSRDP persistence in default mode for a Windows client-sever configuration using RDP.

Configuring MSRDP 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 LTM system to allow the LTM system to consistently connect that client to the same server. Enabling MSRDP persistence to behave in this way is the non-default mode.

The MSRDP profile

To implement MSRDP persistence, you either use the default **msrdp** profile or create a custom profile. Table 9.4 shows the settings and their values that make up the default **msrdp** profile.

Setting	Description	Default Value
Name	Specifies a unique name for the profile. This setting is required.	No default value
Persistence Type	Specifies the type of persistence profile. This setting is required.	**Microsoft Remote Desktop**
Match Across Services	Specifies that all persistent connections from a client IP address that go to the same virtual IP address also go to the same node.	Disabled
Match Across Virtual Servers	Specifies that all persistent connections from the same client IP address go to the same node.	Disabled
Match Across Pools	Specifies that the LTM system can use any pool that contains this persistence entry.	Disabled
Timeout	Specifies the number of seconds before a persistence entry times out.	**0**
Has Session Directory	Specifies whether or not the server is running Session Directory.	**Enabled**

Table 9.4 *Settings of an MSRDP persistence profile*

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

The SIP profile

To implement SIP persistence, you either use the default **sip** profile, or create a custom profile. Table 9.5 shows the settings of the default **sip** profile and their values.

Setting	Description	Default Value
Name	Specifies a unique name for the profile. This setting is required.	No default value
Persistence Type	Specifies the type of persistence profile. This setting is required.	**SIP**
Match Across Services	Specifies that all persistent connections from a client IP address that go to the same virtual IP address also go to the same node.	Disabled
Match Across Virtual Servers	Specifies that all persistent connections from the same client IP address go to the same node.	Disabled
Match Across Pools	Specifies that the LTM system can use any pool that contains this persistence entry.	Disabled
Timeout	Specifies the number of seconds before a persistence entry times out.	**180**

Table 9.5 *Settings of a SIP persistence profile*

◆ Note

The LTM system currently supports persistence for SIP messages sent through UDP only.

The timeout value that you specify allows the LTM 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 180 seconds. If you change the timeout value, we recommend that the value be no lower than the default.

Source address affinity persistence

Source address affinity persistence, also known as simple persistence, tracks sessions based only on the source IP address. When a client requests a connection to a virtual server that supports source address affinity persistence, the LTM system checks to see if that client previously connected, and if so, returns the client to the same pool member.

You might want to use source address affinity persistence and SSL 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 LTM system to direct the client to the original pool member based on the client's IP address. As long as the client's source address affinity persistence record has not timed out, the LTM system can successfully return the client to the appropriate pool member.

Persistence settings 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**.

The persistence mask feature works only for virtual servers that implement source address affinity persistence. By adding a persistence mask, you identify a range of source IP addresses to manage together as a single source address affinity persistent connection when connecting to the pool.

The source address affinity persistence profile

To implement source address affinity persistence, you can either use the default **source_addr** profile or create a custom profile. Table 9.6 shows the settings and values that make up the default **source_addr** profile.

Setting	Description	Default Value
Name	Specifies a unique name for the profile. This setting is required.	No default value
Persistence Type	Specifies the type of persistence profile. This setting is required.	**Source Address Affinity**
Match Across Services	Specifies that all persistent connections from a client IP address that go to the same virtual IP address also go to the same node.	Disabled
Match Across Virtual Servers	Specifies that all persistent connections from the same client IP address go to the same node.	Disabled
Match Across Pools	Specifies that the LTM system can use any pool that contains this persistence entry.	Disabled
Timeout	Specifies the number of seconds before a persistence entry times out.	**180**
Mask	Specifies the mask that the LTM system should use before matching with an existing persistence entry.	**0.0.0.0**
Map Proxies	Enables or disables proxy mapping.	**Enabled**

Table 9.6 *Settings of a source address affinity persistence profile*

SSL persistence

SSL persistence is a type of persistence that tracks SSL sessions 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 LTM system still recognizes the session as being persistent based on the session ID.

You might want to use SSL persistence and source address affinity persistence together. In situations where an SSL session ID times out, or where a returning client does not provide a session ID, you might want the LTM 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 LTM system can successfully return the client to the appropriate node.

◆ Note

The SSL persistence type is only valid for systems that are not performing SSL certificate-based authentication of client requests or server responses. If you are using Client SSL or Server SSL profiles to configure certificate-based authentication, do not configure an SSL persistence profile. Instead, see Chapter 13, ***Writing iRules****.*

The SSL profile

To implement SSL persistence, you either use the default **ssl** profile or create a custom profile. Table 9.7 shows the settings and their values that make up the default profile for SSL persistence.

Setting	Description	Default Value
Name	Specifies a unique name for the profile. This setting is required.	No default value
Persistence Type	Specifies the type of persistence profile. This setting is required.	**SSL**
Match Across Services	Specifies that all persistent connections from a client IP address that go to the same virtual IP address also go to the same node.	Disabled
Match Across Virtual Servers	Specifies that all persistent connections from the same client IP address go to the same node.	Disabled
Match Across Pools	Specifies that the LTM system can use any pool that contains this persistence entry.	Disabled
Timeout	Specifies the number of seconds before a persistence entry times out. That is, this setting sets the SSL session ID timeout value, which determines how long the LTM system stores a given SSL session ID before removing the ID from the system.	**300**

Table 9.7 *Settings of an SSL persistence profile*

Universal persistence

Included in the LTM system's Universal Inspection Engine (UIE) is a set of functions that you can specify within LTM iRules to direct traffic in more granular ways. Using these iRule functions, you can write expressions that direct traffic based on content data, or direct traffic to a specific member of a pool.

Universal persistence takes this iRules feature one step further, by allowing you to implement persistence for sessions based on content data, or based on connections to a specific member of a pool. Universal persistence does this by defining some sequence of bytes to use as a session identifier.

To use iRule expressions for persistence, a universal persistence profile includes a setting that specifies the name of the iRule containing the expression.

Once you have created an iRule and specified the iRule name within a universal profile, you must assign both the iRule and the profile to the appropriate virtual server as resources.

For a complete description of iRule expressions and functions, see Chapter *13, Writing iRules.*

The universal persistence profile

To implement universal persistence, you can either use the default **universal** profile or create a custom profile. Table 9.8 shows the settings and values that make up the default profile for universal persistence.

Setting	Description	Default Value
Name	Specifies a unique name for the profile. This setting is required.	No default value
Persistence Type	Specifies the type of persistence. This setting is required.	**Universal**
Match Across Services	Specifies that all persistent connections from a client IP address that go to the same virtual IP address also go to the same node.	Disabled
Match Across Virtual Servers	Specifies that all persistent connections from the same client IP address go to the same node.	Disabled
Match Across Pools	Specifies that the LTM system can use any pool that contains this persistence entry.	Disabled
iRule	Specifies the name of an existing iRule that the LTM should run to determine a persistence entry.	**_sys_auth_ldap**
Timeout	Specifies the number of seconds before a persistence entry times out.	**180**

Table 9.8 *Settings of a universal persistence profile*

10

Configuring Monitors

- Introducing monitors
- Creating a custom monitor
- Configuring monitor settings
- Special configuration considerations
- Associating monitors with pools and nodes
- Managing monitors

Introducing monitors

An important feature of the BIG-IP® local traffic management (LTM) system is a load-balancing tool called monitors. ***Monitors*** verify connections on pool members and nodes. A monitor can be either a health monitor or a performance monitor, designed to check the status of a pool, pool member, or node on an ongoing basis, at a set interval. If a pool member or node being checked does not respond within a specified timeout period, or the status of a pool member or node indicates that performance is degraded, the LTM system can redirect the traffic to another pool member or node.

Some monitors are included as part of the LTM system, while other monitors are user-created. Monitors that the LTM system provides are called ***pre-configured monitors***. User-created monitors are called ***custom monitors***. For more information on pre-configured and custom monitors, see *Understanding pre-configured and custom monitors*, on page 10-6.

Before configuring and using monitors, it is helpful to understand some basic concepts regarding monitor types, monitor settings, and monitor implementation.

- **Monitor types**
  Every monitor, whether pre-configured or custom, is a certain type of monitor. Each type of monitor checks the status of a particular protocol, service, or application. For example, one type of monitor is HTTP. An HTTP type of monitor allows you to monitor the availability of the HTTP service on a pool, pool member, or node. A WMI type of monitor allows you to monitor the performance of a pool, pool member, or node that is running the Windows Management Instrumentation (WMI) software. An ICMP type of monitor simply determines whether the status of a node is **up** or **down**. For more information on monitor types, see *Summary of monitor types*, on page 10-2 and *Configuring monitor settings*, on page 10-10.
- **Monitor settings**
  Every monitor consists of settings with values. The settings and their values differ depending on the type of monitor. In some cases, the LTM system assigns default values. For example, Figure 10.1 shows that an ICMP-type monitor has these settings and default values.

```
Name my_icmp
Type ICMP
Interval 5
Timeout 16
Transparent No
Alias Address * All Addresses
```

Figure 10.1 *Example of a monitor with default values*

The settings in Figure 10.1 specify that an ICMP type of monitor is configured to check the status of an IP address every 5 seconds, and to time out every 16 seconds. The destination IP address that the monitor

checks is specified by the **Alias Address** setting, with the value *** All Addresses**. Thus, in the preceding example, all IP addresses with which the monitor is associated are checked. For more information on monitor settings, see *Summary of monitor settings*, on page 10-4 and *Configuring monitor settings*, on page 10-10.

- **Monitor implementation**
  The task of implementing a monitor varies depending on whether you are using a pre-configured monitor or creating a custom monitor. If you want to implement a pre-configured monitor, you need only associate the monitor with a pool, pool member, or node. If you want to implement a custom monitor, you must first create the custom monitor, and then associate it with a pool, pool member, or node. For more information on implementing a monitor, see *Understanding pre-configured and custom monitors*, on page 10-6 and *Creating a custom monitor*, on page 10-9.

Summary of monitor types

The LTM system includes many different types of monitors, each designed to perform a specific type of monitoring. Table 10.1 lists the monitor types that you can configure for controlling your network traffic.

Monitor Types	Description
Simple monitors	
ICMP	Checks the status of a node, using Internet Control Message Protocol (ICMP).
Gateway ICMP	Checks nodes in a pool that implements gateway failsafe for high availability.
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.
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 an LTM system. Once downloaded successfully, the file is not saved.

Table 10.1 *Monitor types available on an LTM system*

Monitor Types	Description
IMAP	Verifies the Internet Message Access Protocol (IMAP) by attempting to open a specified mail folder on a server. This monitor is similar to the **pop3** monitor.
LDAP	Verifies the Lightweight Directory Access Protocol (LDAP) service by attempting to authenticate the specified user.
MSSQL	Verifies Microsoft® Windows SQL-based services.
NNTP	Verifies the Usenet News protocol (NNTP) service by attempting to retrieve a newsgroup identification string from the server.
Oracle	Verifies services based on Oracle® by attempting to perform an Oracle login to a service.
POP3	Verifies the Post Office Protocol (pop3) service by attempting to connect to a pool, pool member, or node, log on as the specified user, and log off.
RADIUS	Verifies the Remote Access Dial-in User Service (RADIUS) service by attempting to authenticate the specified user.
Real Server	Checks the performance of a pool, pool member, or node that is running the RealServer data collection agent, and then dynamically load balances traffic accordingly.
SIP	Checks the status of Session Initiation Protocol (SIP) Call-ID services on a device. The SIP protocol enables real-time messaging, voice, data, and video.
SMTP	Checks the status of a pool, pool member, or node by issuing standard Simple Mail Transport Protocol (SMTP) commands.
SNMP DCA	Checks the current CPU, memory, and disk usage of a pool, pool member, or 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 pool, pool member, or node that is running an SNMP data collection agent, and then dynamically load balances traffic accordingly. The way that you configure the monitor settings determines the data that the LTM system collects.
SOAP	Tests a Web service based on the Simple Object Access Protocol (SOAP).
UDP	Verifies the User Datagram Protocol (UDP) service by attempting to send UDP packets to a pool, pool member, or node and receiving a reply.
WMI	Checks the performance of a pool, pool member, or node that is running the Windows Management Infrastructure (WMI) data collection agent and then dynamically load balances traffic accordingly.

Table 10.1 *Monitor types available on an LTM system*

Summary of monitor settings

Monitors contain settings with corresponding values. These settings and their values affect the way that a monitor performs its status check. When you create a custom monitor, you must configure these setting values. For those settings that have default values, you can either retain the default values, or modify them to suit your needs.

Table 10.2 contains a complete list of all possible monitor settings, along with their descriptions. Note that each monitor contains only a subset of these settings.

Setting	Definition
Additional Accepted Status Codes	Status codes for a SIP monitor. Acceptable values are **Any**, **None**, or a list of status codes that you specify.
Agent	An agent specification for use with Real Server, SNMP Base, and WMI monitors only.
Agent Type	The type of agent running on the server that you are monitoring with an SNMP DCA monitor.
Alias Address	The destination node for a ping. Usually has a value of *****, which checks all nodes. This setting causes the monitor instance to ping the IP address for which it is instantiated. Specifying an individual IP address forces the destination to that address (that is, a node). When the monitor includes the **Alias Address** setting but not the **Alias Service Port** setting, the monitor pings a node address rather than a pool member.
Alias Service Port	The destination pool member for a ping. Usually has a value of ***:***, which checks all pool members. This setting causes the monitor instance to ping the IP address and port for which it is instantiated. Specifying an individual IP address and port forces the destination to that address and port (that is, a pool member).
Arguments	Any command-line arguments that are required.
Base	Starting place in the LDAP hierarchy from which to begin the query, for LDAP monitors only.
Cipher List	List of ciphers, for use with HTTPS monitors only.
Command	A command associated with metrics and metric values. Applies to Real Server and WMI monitors.
Community	A setting that applies to SNMP DCA monitors only. Default value is **Public**.
CPU Coefficient	A CPU value used for calculating a ratio weight.
CPU Threshold	The highest CPU threshold value allowed, used in calculating a ratio weight.
Database	Database name, for Oracle and MSSQL monitors only.
Disk Coefficient	A disk value used for calculating a ratio weight.
Disk Threshold	The highest disk threshold value allowed, used in calculating a ratio weight.

Table 10.2 *List of all possible monitor settings*

Setting	Definition
Domain	Domain name, for SMTP monitors only.
External Program	A user-created monitor type.
Filter	LDAP-format key of what is to be searched for, for LDAP monitors only.
Folder	Folder name for IMAP monitors only.
Interval	Time interval for frequency of pool, pool member, or node checking, in seconds.
Memory Coefficient	A memory value used for calculating a ratio weight.
Memory Threshold	The highest disk threshold value allowed, used in calculating a ratio weight.
Method	A method specification such as **GET** or **POST**. Applies to Real Server, SOAP, and WMI monitors only.
Metrics	Metrics that you want to monitor, such as CPU percentage or memory usage. Applies to Real Server and WMI monitors only.
Mode	Mode of the monitor.
Newsgroup	Newsgroup, for NNTP monitors only.
Password	Password for services with password security.
Path/Filename	For FTP monitors only, this setting **r**eplaces the **Send String** setting**.** You can use this setting to specify a full path to a file.
Receive Row	The row in the returned table that contains the **Receive String** value.
Receive Column	The column in the returned table that contains the **Receive String** value.
Receive String	Receive expression for ECV checking. Default **Send String** and **Receive String** values are empty (""), which match any string.
Reverse	A mode that sets the pool, pool member, or node to a **down** state if the received content matches the **Receive String** string.
Secret	Shared secret for RADIUS monitors only.
Security	The security protocol that the monitor should use (**SSL**, **TLS**, or **None**). Applies to LDAP monitors only.
Send Packets	Number of packets to send when using the UDP monitor.
Send String	Send string for ECV checking. Default **Send String** and **Receive String** values are empty (""), which matches any string.
Timeout	Timeout for checking a pool, pool member, or node, in seconds.
Timeout Packets	Timeout in seconds for receiving UDP packets.

Table 10.2 *List of all possible monitor settings*

Setting	Definition
Transparent	A mode that forces pinging through the pool, pool member, or node to the IP address and/or port for transparent pool members and nodes, such as firewalls.
URL	For a WMI monitor, supplies a URL.
URL Path	For a SOAP monitor, supplies a URL path.
User Name	User name for services with password security. For LDAP monitors only, this is a distinguished name, that is, LDAP-format user name.

Table 10.2 *List of all possible monitor settings*

Understanding pre-configured and custom monitors

When you want to monitor the health or performance of pool members or nodes, you can either use a pre-configured monitor, or create and configure a custom monitor.

Using pre-configured monitors

For a subset of monitor types, the LTM system includes a set of pre-configured monitors. A ***pre-configured monitor*** is an existing monitor that the LTM system provides for you, with its settings already configured. You cannot modify pre-configured monitor settings, as they are intended to be used as is. The purpose of a pre-configured monitor is to eliminate the need for you to explicitly create one. You use a pre-configured monitor when the values of the settings meet your needs as is.

The names of the pre-configured monitors that the LTM includes are:

- **gateway_icmp**
- **http**
- **https**
- **https_443**
- **icmp**
- **real_server**
- **snmp_dca**
- **tcp**
- **tcp_echo**
- **tcp_half_open**

An example of a pre-configured monitor is the **icmp** monitor. Figure 10.2 shows the **icmp** monitor, with values configured for its **Interval**, **Timeout**, and **Alias Address** settings. Note that the **Interval** value is **5**, the **Timeout** value is **16**, the Transparent value is **No**, and the **Alias Address** value is *** All Addresses**.

```
Name icmp
Type ICMP
Interval 5
Timeout 16
Transparent No
Alias Address * All Addresses
```

***Figure 10.2** The **icmp** pre-configured monitor*

If the **Interval**, **Timeout**, **Transparent**, and **Alias Address** values meet your needs, you simply assign the **icmp** pre-configured monitor directly to a pool, pool member, or node, using the **Pools** or **Nodes** screens within the Configuration utility. In this case, you do not need to use the Monitors screens, unless you simply want to view the values of the pre-configured monitor settings.

If you do not want to use the values configured in a pre-configured monitor, you can create a custom monitor.

Using custom monitors

A ***custom monitor*** is a monitor that you create based on one of the allowed monitor types.You create a custom monitor when the values defined in a pre-configured monitor do not meet your needs, or no pre-configured monitor exists for the type of monitor you are creating. (For information on monitor types, see *Summary of monitor types*, on page 10-2.)

Importing settings from a pre-configured monitor

If a pre-configured monitor exists that corresponds to the type of custom monitor you are creating, you can import the settings and values of that pre-configured monitor into the custom monitor. You are then free to change those setting values to suit your needs. For example, if you create a custom monitor called **my_icmp**, the monitor can inherit the settings and values of the pre-configured monitor **icmp**. This ability to import existing setting values is useful when you want to retain some setting values for your new monitor but modify others.

Figure 10.3 shows an example of a custom ICMP-type monitor called **my_icmp**, which is based on the pre-configured monitor **icmp**. Note that the Interval value has been changed to **10**, and the **Timeout** value to **20**. The other settings retain the values defined in the pre-configured monitor.

```
Name my_icmp
Type ICMP
Interval 10
Timeout 20
Transparent No
Alias Address * All Addresses
```

Figure 10.3 *A custom monitor based on a pre-configured monitor*

Importing settings from a custom monitor

You can import settings from another custom monitor instead of from a pre-configured monitor. This is useful when you would rather use the setting values defined in another custom monitor, or when no pre-configured monitor exists for the type of monitor you are creating. For example, if you create a custom monitor called **my_oracle_server2**, you can import settings from an existing Oracle-type monitor such as **my_oracle_server1**. In this case, because the LTM system does not provide a pre-configured Oracle-type monitor, a custom monitor is the only kind of monitor from which you can import setting values.

Selecting a monitor is straightforward. Like **icmp**, each of the monitors has a **Type** setting 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 monitors, like the **external** monitor, which calls a user-supplied program.)

For procedures on selecting and configuring a monitor, see *Creating a custom monitor*, on page 10-9.

Importing settings from a monitor template

If no pre-configured or custom monitor exists that corresponds to the type of monitor you are creating, the LTM system imports settings from a monitor template. A ***monitor template*** is a non-visible entity that exists within the LTM system for each monitor type and contains a group of settings and default values. A monitor template merely serves as a tool for the LTM system to use for importing settings to a custom monitor when no monitor of that type already exists.

Creating a custom monitor

When you create a custom monitor, you use the Configuration utility to: give the monitor a unique name, specify a monitor type, and, if a monitor of that type already exists, import settings and their values from the existing monitor. You can then change the values of any imported settings.

You must base each custom monitor on a monitor type. When you create a monitor, the Configuration utility displays a list of monitor types. To specify a monitor type, simply choose the one that corresponds to the service you want to check. For example, if you want to want to create a monitor that checks the health of the HTTP service on a pool, you choose **HTTP** as the monitor type.

If you want to check more than one service on a pool or pool member (for example **HTTP** and **HTTPS)**, you can associate more than one monitor on that pool or pool member. For more information, see Chapter 4, *Configuring Load Balancing Pools*.

Checking services is not the only reason for implementing a monitor. 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 monitors, **icmp** or **tcp_echo**. Or, if you want to verify TCP only, use the monitor **tcp**.

◆ Note

Before creating a custom monitor, you must decide on a monitor type. For information on monitor types, see ***Configuring monitor settings****, on page 10-10.*

To create a custom monitor

1. On the Main tab, expand **Local Traffic**.
   The Local Traffic menu expands.
2. Click **Monitors**.
   This displays a list of existing monitors.
3. In the upper right corner of the screen, click **Create**.
   The New Monitor screen opens.
4. For the **Type** setting, select the type of monitor that you want to create.
   If a monitor of that type already exists, **Import Settings** appears.
5. Specify a name, based on one of these settings:
   - If **Import Settings** appears, choose a monitor name from the list.
   - If a monitor of the type you selected does not exist, in the **Name** box, type a unique name for the custom monitor.
6. In the Configuration section of the screen, select **Advanced**. This allows you to modify additional default settings.
7. Configure all settings shown.
8. Click **Finished**.

Configuring monitor settings

Before you can create a custom monitor, you must select a monitor type. Monitors types fall into three categories:

- **Simple monitors**
  These are health monitors that monitor the status of a node.
- **Extended Content Verification (ECV) monitors**
  These are health monitors that verify service status by retrieving specific content from pool members or nodes.
- **External Application Verification (EAV) monitors**
  These are health or performance monitors that verify service status by executing remote applications, using an external service-checker program.

Simple monitors

Simple monitors are those that check nodes only, and not pool members. The simple monitor types are:

- ICMP
- Gateway ICMP
- TCP Echo
- TCP Half Open

The LTM system provides a set of pre-configured simple monitors: **icmp**, **gateway_icmp**, **tcp_echo**, and **tcp_half_open**. You can either use these pre-configured monitors as is, or create custom monitors of these types.

The following sections describe each type of simple monitor and show the pre-configured monitor for each type. Note that each pre-configured monitor consists of settings and their values. The boldfaced type within each pre-configured monitor serves to distinguish the settings from their corresponding values.

Important

When defining values for custom monitors, make sure you avoid using any values that are on the list of reserved keywords. For more information, see solution number 3653 (for 9.0+ systems) on the Ask F5 web site.

ICMP

Using an ICMP type of monitor, you can use Internet Control Message Protocol (ICMP) to make a simple node check. The check is successful if the monitor receives a response to an ICMP_ECHO datagram. Figure 10.4 shows the settings and their values for the pre-configured monitor **icmp**.

```
Name icmp
Type ICMP
Interval 5
Timeout 16
Transparent No
Alias Address * All Addresses
```

Figure 10.4 *The **icmp** pre-configured monitor*

The **Transparent** mode is an option for ICMP-type monitors. When you set this mode to **Yes**, the monitor pings the node with which the monitor is associated. For more information about **Transparent** mode, refer to *Using transparent and reverse modes*, on page 10-37.

Gateway ICMP

A Gateway ICMP type of monitor has a special purpose. You use this monitor for a pool that implements gateway failsafe for high availability.

A Gateway ICMP monitor functions the same way as an ICMP monitor, except that you can apply a Gateway ICMP monitor to a pool member. (Remember that you can apply an ICMP monitor to a node only and not a pool member.) Figure 10.5 shows the settings and their values for the pre-configured **gateway_icmp** monitor.

```
Name gateway_icmp
Type Gateway ICMP
Interval 5
Timeout 16
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.5 *The **gateway_icmp** pre-configured monitor*

TCP Echo

With a TCP Echo type of monitor, you can verify Transmission Control Protocol (TCP) connections. The check is successful if the LTM system receives a response to a TCP Echo message. The TCP Echo type 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 10-37.)

To use a TCP Echo monitor type, you must ensure that TCP Echo is enabled on the nodes being monitored. Figure 10.6 shows the settings for the pre-configured monitor **tcp_echo**.

```
Name tcp_echo  {
Type TCP Echo
Interval 5
Timeout 16
Transparent No
Alias Address * All Addresses
```

***Figure 10.6** The **tcp_echo** pre-configured monitor*

TCP Half Open

A TCP Half Open type of monitor performs a quick check on the associated service by sending a TCP SYN packet to the service. As soon as the monitor receives the SYN-ACK packet from the service, the monitor considers the service to be in an **up** state, and sends a RESET to the service instead of completing the three-way handshake.

Figure 10.7 shows the settings for the pre-configured monitor **tcp_half_open**.

```
Name tcp_half_open  {
Type TCP Half Open
Interval 5
Timeout 16
Transparent No
Alias Addresses * All Addresses
Alis Service Ports * All Ports
```

***Figure 10.7** The **tcp_half_open** pre-configured monitor*

Extended Content Verification (ECV) monitors

ECV monitors use **Send String** and **Receive String** settings in an attempt to retrieve explicit content from pool members or nodes. The LTM system provides the pre-configured monitors **tcp**, **http**, **https**, and **https_443** for these ECV monitor types:

- TCP
- HTTP
- HTTPS

You can associate TCP, HTTP, and HTTPS monitors with pools and pool members only. You cannot associate them with nodes. You can either use the pre-configured ECV monitors as is, or create custom monitors from these monitor types.

The following sections describe each type of simple monitor and show the pre-configured monitor for each type. Note that each pre-configured monitor consists of settings and their values. The boldfaced type within each pre-configured monitor serves to distinguish the settings from their corresponding values.

Important

When defining values for custom monitors, make sure you avoid using any values that are on the list of reserved keywords. For more information, see solution number 3653 (for 9.0+ systems) on the Ask F5 web site.

TCP

A TCP type of monitor attempts to receive specific content sent over TCP. The check is successful when the content matches the **Receive String** value. A TCP type of monitor takes a **Send String** value and a **Receive String** value. If the **Send String** value is blank and a connection can be made, the service is considered **up**. A blank **Receive String** value matches any response. Both **Transparent** and **Reverse** modes are options. For more information about **Transparent** and **Reverse** modes, see *Using transparent and reverse modes*, on page 10-37.

Figure 10.8 shows the settings for the pre-configured monitor **tcp**.

```
Name tcp
Type TCP
Interval 5
Timeout 16
Send String ""
Receive String ""
Reverse No
Transparent No
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.8 *The **tcp** pre-configured monitor*

Important

*When specifying **Send String** and **Receive String** values, do not use the **Enter** key to specify a new line when more than one line is required for the string. Instead, type the escape characters **\n**.*

HTTP

You can use an HTTP type of monitor to check the status of 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, may send a user name and password. The check is successful when the content matches the **Receive String** value. An HTTP monitor uses a

send string, a receive string, a user name, a password, and optional **Reverse** and **Transparent** modes. (If there is no password security, you must use blank strings [""] for the **User Name** and **Password** settings.)

For more information on **transparent** and **reverse** modes, see *Using transparent and reverse modes*, on page 10-37.

Figure 10.9 shows the settings of the pre-configured monitor **http**.

```
Name http
Type HTTP
Interval 5
Timeout 16
Send String GET /
Receive String ""
User Name ""
Password ""
Reverse No
Transparent No
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.9 *The **http** pre-configured monitor*

Important

*When specifying **Send String** and **Receive String** values, do not use the **Enter** key to specify a new line when more than one line is required for the string. Instead, type the escape characters **\n**.*

HTTPS

You use an HTTPS type of monitor to check the status of Hypertext Transfer Protocol Secure (HTTPS) traffic. An HTTPS type of monitor attempts to receive specific content from a web page protected by SSL security. The check is successful when the content matches the **Receive String** value.

HTTPS-type monitors use a send string, a receive string, a user name, a password, and an optional **Reverse** setting. (If there is no password security, you must use blank strings [""] for the **Username** and **Password** settings.) For more information on the **Reverse** setting, see *Using transparent and reverse modes*, on page 10-37.

Important

*When specifying **Send String** and **Receive String** values, do not use the **Enter** key to specify a new line when more than one line is required for the string. Instead, type the escape characters **\n**.*

HTTP-type monitors also include the settings **Cipher List**, **Compatibility**, and **Client Certificate**. If you do not specify a cipher list, the monitor uses the default cipher list **DEFAULT:+SHA:+3DES:+kEDH**. When you set

the **Compatibility** setting to **Enabled**, this sets the SSL options to ALL. You use the **Client Certificate** setting to specify a certificate file that the monitor then presents to the server.

The LTM system provides two pre-configured HTTPS monitors, **https** and **https_443**. Figure 10.10 shows the settings of the pre-configured monitor **https**, and Figure 10.11 shows the settings of the pre-configured **https_443**.

```
Name https
Type HTTPS
Interval 5
Timeout 16
Send String GET /
Receive String ""
Cipher List ""
User Name ""
Password ""
Compatibility Enabled
Client Certificate ""
Reverse No
Alias Address * All Addresses
Alias Service Port * All Ports
```

***Figure 10.10** The **https** pre-configured monitor*

.

```
Name https_443
Type HTTPS_443
Interval 5
Timeout 16
Send String GET /
Receive String ""
Cipher List ""
User Name ""
Password ""
Compatibility Enabled
Client Certificate ""
Reverse No
Alias Address * All Addresses
Alias Service Port HTTPS
```

***Figure 10.11** The **https_443** pre-configured monitor*

The **Reverse** mode is an option for monitors that import settings from the **https** and **https_443** monitors. For more information on **Reverse** mode, see *Using transparent and reverse modes*, on page 10-37.

External Application Verification (EAV) monitors

EAV monitors verify applications on servers by running those applications remotely, using an external service checker program located in the directory **/user/bin/monitors**.

The types of EAV monitors that you can create are:

- External
- FTP
- IMAP
- LDAP
- MSSQL
- NNTP
- Oracle
- POP3
- RADIUS
- Real Server
- Scripted
- SIP
- SMTP
- SNMP DCA
- SNMP DCA Base
- SOAP
- UDP
- WAP
- WMI

The LTM system provides two pre-configured EAV monitors, **snmp_dca** and **real_server**, based on the monitor types SNMP DCA and Real Server. For any other EAV monitor type that you want to use, you create a custom monitor.

The following sections describe each type of simple monitor and show the pre-configured monitor for each type. Note that each pre-configured monitor consists of settings and their values. The boldfaced type within each pre-configured monitor serves to distinguish the settings from their corresponding values.

Important

When defining values for custom monitors, make sure you avoid using any values that are on the list of reserved keywords. For more information, see solution number 3653 (for 9.0+ systems) on the Ask F5 web site.

External

Using an External type of monitor, you can create your own monitor type. To do this, you create a custom External-type monitor and within it, specify a user-supplied monitor to run.

It is the **External Program** setting that you use to specify the executable name of your user-supplied monitor program. By default, an External-type monitor searches the directory **/user/bin/monitors** for that monitor name. If the user-supplied monitor resides elsewhere, you must enter a fully qualified path name.

The **Arguments** setting allows you to specify any command-line arguments that are required.

Figure 10.12 shows the settings and default values of an External-type monitor.

```
Name my_external
Type External
Interval 5
Timeout 16
External Program ""
Arguments ""
Variables ""
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.12 *An External-type custom monitor with default values*

FTP

Using an FTP type of monitor, you can monitor File Transfer Protocol (FTP) traffic. A monitor of this type attempts to download a specified file to the **/var/tmp** directory, and if the file is retrieved, the check is successful. Note that once the file has been successfully downloaded, the LTM system does not save it.

An FTP monitor specifies a user name, a password, and a full path to the file to be downloaded.

Figure 10.13 shows the settings and default values of an FTP-type monitor.

```
Name my_ftp
Type FTP
Interval 10
Timeout 31
User Name ""
Password ""
Path/Filename ""
Mode Passive
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.13 *An FTP-type custom monitor with default values*

IMAP

With an IMAP type of monitor, you can check the status of Internet Message Access Protocol (IMAP) traffic. An IMAP monitor is essentially a POP3 type of monitor with the addition of the **Folder** setting. The check is successful if the monitor is able to log into a server and open the specified mail folder.

An IMAP monitor requires that you specify a user name and password. Figure 10.14 shows the settings and default values of an IMAP-type monitor.

```
Name my_imap
Type IMAP
Interval 5
Timeout 16
User Name ""
Password ""
Folder INBOX
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.14 *An IMAP-type custom monitor with default values*

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

An LDAP type of monitor checks the status of Lightweight Directory Access Protocol (LDAP) servers. The LDAP protocol 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 user name, a password, and base and filter strings. Figure 10.15 shows the settings and default values of an LDAP-type monitor.

```
Name my_ldap
Type LDAP
Interval 10
Timeout 31
User Name ""
Password ""
Base ""
Filter ""
Security None
Mandatory Attributes No
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.15 *An LDAP-type custom monitor with default values*

The **User Name** setting specifies a distinguished name, that is, an LDAP-format user name.

The **Base** setting specifies the starting place in the LDAP hierarchy from which to begin the query.

The **Filter** setting specifies an LDAP-format key of the search item.

The **Security** setting specifies the security protocol to be used. Acceptable values are **SSL**, **TLS**, or **None**.

The **Mandatory Attributes** setting affects the way that the system conducts the filter search. When the value is **No**, the system performs a one-level search for attributes, and if the search returns no attributes, the node is reported as **up**. When the value is **Yes**, the system performs a subtree search, and if the search returns no attributes, the node is not reported as **up**.

For an LDAP monitor to work properly, the BIG-IP system must be able to perform a reverse DNS lookup on the address of the LDAP or LDAPS node. This reverse lookup allows the BIG-IP system to check the host name of the node's address when it verifies the SSL certificate. An external DNS server does not work with this type of monitor

The reverse DNS lookup requirement applies to both LDAP and LDAPS nodes, even though LDAP does not require the use of an SSL certificate.

◆ Note

*You do not need to insert an entry for each LDAP server into the **/etc/hosts** file.*

MSSQL

You use an MSSQL type of monitor to perform service checks on Microsoft SQL Server-based services such as Microsoft SQL Server versions 6.5 and 7.0.

The LTM system requires installation of a JDBC driver before performing the actual login. For more information, see Appendix A, *Additional Monitor Considerations*.

If you receive a message that the connection was refused, verify that the IP address and port number or service are correct. If you are still having login trouble, see *Troubleshooting MSSQL logins*, on page 10-21.

The remainder of this section on MSSQL monitors describes prerequisite tasks, the default monitor settings, and troubleshooting tips.

Prerequisite tasks for MSSQL

Before using an MSSQL-type monitor, you must download a set of JDBC Java™ Archive (JAR) files and install them on the LTM system. For more information, see Appendix A, *Additional Monitor Considerations*.

MSSQL monitor settings and their default values

Figure 10.16 shows the settings and default settings of an MSSQL-type monitor.

```
Name my_mssql
Type mssql
Interval 30
Timeout 91
Send String ""
Receive String ""
User Name ""
Password ""
Database ""
Receive Row ""
Receive Column ""
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.16 *An MSSQL-type custom monitor with default values*

Important

*When specifying **Send String** and **Receive String** values, do not use the **Enter** key to specify a new line when more than one line is required for the string. Instead, type the escape characters **\n**.*

In an MSSQL-type monitor, the **Database** setting specifies the name of the data source on the Microsoft® SQL-based server. Examples are **sales** and **hr**.

The **Send String** setting is optional and specifies a SQL query statement that the LTM system should send to the server. Examples are **SELECT * FROM sales** and **SELECT FirstName, LastName From Employees**. If you configure the **Send String** setting, you can also configure these settings:

- **Receive String**
  The **Receive String** setting is an optional parameter that specifies the value expected to be returned for the row and column specified with the **Receive Row** and **Receive Column** settings. An example of a **Receive String** value is **ALAN SMITH**. You can only configure this setting when you configure the **Send String** setting.
- **Receive Row**
  The **Receive Row** setting is optional, and is useful only if the **Receive String** setting is specified. This setting specifies the row in the returned table that contains the **Receive String** value. You can only configure this setting when you configure the **Send String** setting.
- **Receive Column**
  The **Receive Column** setting is optional and is useful only if the **Receive String** setting is specified. This setting specifies the column in the returned table that contains the **Receive String** value. You can only configure this setting when you configure the **Send String** setting.

Troubleshooting MSSQL logins

If an MSSQL monitor cannot log in to the server, and you have checked that the specified IP address and port number or service are correct, try the following:

- **Verify that you can log in using another tool**.
  For example, the server program Microsoft NT SQL Server version 6.5 includes a client program named ISQL/w. This client program performs simple logins to SQL servers. Use this program to test whether you can log in to the server using the ISQL/w program.
- **Add login accounts using the Microsoft SQL Enterprise Manager**.
  On the Microsoft SQL Server, you can run the SQL Enterprise Manager to add login accounts. When first entering the SQL Enterprise Manager, you may be prompted for the SQL server that you want to manage.

  You can register servers by entering the machine name, user name, and password. If these names are correct, the server becomes registered and you are then able to click an icon for the server. When you expand the subtree for the server, there is an icon for login accounts.

  Beneath 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 access the **Add login** option. 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.

NNTP

You use an NNTP type of monitor to check the status of 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 user name and password.

Figure 10.17 shows the settings and default values of an NNTP-type monitor.

```
Name my_nntp
Type NNTP
Interval 5
Timeout 16
User Name ""
Password ""
Newsgroup ""
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.17 *An NNTP-type custom monitor with default values*

Oracle

With an Oracle type of monitor, you can check the status of an Oracle database server. The check is successful if the monitor is able to connect to the server, log in as the indicated user, and log out.

Figure 10.18 shows the settings and default values of an Oracle-type monitor.

```
Name my_oracle
Type Oracle
Interval 30
Timeout 91
Send String GET /
Receive String ""
User Name ""
Password ""
Database ""
Receive Row ""
Receive Column ""
Alias Address * All Addresses
Alias Service Port * All Ports
```

***Figure 10.18** An Oracle-type custom monitor with default values*

The **Send String** setting specifies a SQL statement that the LTM system should send to the Oracle server. An example is **SELECT * FROM sales**.

The **Receive String** setting is an optional parameter that specifies the value expected to be returned for a specific row and column of the table that the **Send String** setting retrieved. An example of a **Receive String** value is **SMITH**.

Important

*When specifying **Send String** and **Receive String** values, do not use the **Enter** key to specify a new line when more than one line is required for the string. Instead, type the escape characters **\n**.*

In an Oracle type of monitor, the **Database** setting specifies the name of the data source on the Oracle server. Examples are **sales** and **hr**.

The **Receive Row** setting is optional, and is useful only if the **Receive String** setting is specified. This setting specifies the row in the returned table that contains the **Receive String** value.

The **Receive Column** setting is optional and is useful only if the **Receive String** setting is specified. This setting specifies the column in the returned table that contains the **Receive String** value.

POP3

A POP3 type of monitor checks the status of 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. A POP3 monitor requires a user name and password.

Figure 10.19 shows the settings and default values of a POP3-type monitor

```
Name my_pop3
Type POP3
Interval 5
Timeout 16
User Name ""
Password ""
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.19 *A POP3-type custom monitor with default values*

RADIUS

Using a RADIUS type of monitor, you can check the status of Remote Access Dial-in User Service (RADIUS) servers. The check is successful if the server authenticates the requesting user. A RADIUS monitor requires a user name, a password, and a shared secret string 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.

Figure 10.20 shows the settings and default values of a RADIUS-type monitor.

```
Name my_radius
Type RADIUS
Interval 10
Timeout 31
User Name ""
Password ""
Secret ""
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.20 *A RADIUS-type custom monitor with default values*

Real Server

A Real Server type of monitor checks the performance of a pool, pool member, or node that is running the RealSystem Server data collection agent. The monitor 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, *Configuring Load Balancing Pools* and Appendix A, *Additional Monitor Considerations*.

Note

Unlike health monitors, performance monitors do not report on the status of a pool, pool member, or node.

The LTM system provides a pre-configured Real Server monitor named **real_server**. Figure 10.21 shows the settings and default values of the **real_server** monitor.

```
Name real_server
Type Real Server
Interval 5
Timeout 16
Method GET
Command GetServerStats
Metrics ServerBandwidth:1.5, CPUPercentUsage, MemoryUsage,
   TotalClientCount
Agent Mozilla/4.0 (compatible: MSIE 5.0; Windows NT)
Alias Address * All Addresses
Alias Service Port 12345
```

Figure 10.21 *The **real_server** pre-configured monitor*

Like all pre-configured monitors, the **real_server** monitor is not user-modifiable. However, if you want to modify the **Metrics** setting, you can create a custom Real Server monitor, to which you can add metrics and modify metric values.

Note

*When creating a custom Real Server monitor, you cannot modify the values of the **Method**, **Command**, and **Agent** settings.*

Table 10.3 shows the complete set of server-specific metrics and metric setting default values that apply to the **GetServerStats** command.

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

Table 10.3 *Metrics for a Real Server monitor*

Metric	Default Coefficient	Default Threshold
PNAClientCount	1.0	500
UDPTransportCount	1.0	500
TCPTransportCount	1.0	500
MulticastTransportCount	1.0	500

Table 10.3 *Metrics for a Real Server monitor*

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 10.3 are *metric* defaults, not *monitor* defaults. The monitor defaults take precedence over the metric defaults, just as user-specified values in the custom **real_server** monitor take precedence over the monitor defaults. For example, the monitor shown specifies a coefficient value of **1.5** for **ServerBandwidth** and no value for the other metrics. This means that the monitor uses the monitor 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 monitor default.

Metric coefficient and threshold are the only non-monitor defaults. If a metric not in the monitor is to be added to the custom monitor, it must be added to the list of metrics for the **Metrics** setting. The syntax for specifying non-default coefficient or threshold values is:

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

Scripted

You use the Scripted type of monitor to generate a simple script that reads a file that you create. The file contains **send** and **expect** strings to specify lines that you want to send or that you expect to receive. For example, Figure 10.22 shows a sample file that you could create, which specifies a simple SMTP sequence. Note that the lines of the file are always read in the sequence specified.

```
expect 220
send "HELO bigip1.somecompany.net\r\n"
expect "250"
send "quit\r\n
```

Figure 10.22 *A sample file specifying an SMTP sequence*

Using a Scripted monitor, you can then generate a script that acts on the above file. When the Scripted monitor script reads this file, the script examines each line, and if the line has no double quotes, the line is sent or expected to be received as is. If the line is surrounded by quotation marks, the script strips off the quotation marks, and examines the line for escape characters, treating them accordingly.

Figure 10.23 shows the settings and default values of a Scripted-type monitor.

```
Name my_scripted_monitor
Type Scripted
Import Settings scripted
Interval 10
Timeout 31
Filename <filename>
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.23 *A SIP-type custom monitor with default values*

◆ Note

When you create a file containing send and expect strings, store the file in the directory ***/config/eav****.*

SIP

You use a SIP type of monitor to check the status of SIP Call-ID services. This monitor type uses UDP to issue a request to a server device. The request is designed to identify the options that the server device supports. If the proper request is returned, the device is considered to be **up** and responding to commands.

Figure 10.24 shows the settings and default values of a SIP-type monitor.

```
Name my_sip
Type SIP
Interval 5
Timeout 16
Mode UDP
Additional Accepted Status Codes None
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.24 *A SIP-type custom monitor with default values*

Possible values for the **Mode** setting are **TCP** and **UDP**.

Possible values for the **Additional Accepted Status Codes** setting are **Any**, **None**, and **Status Code List**. The **Status Code List** setting specifies one or more status codes, in addition to status code **200**, that are acceptable in order to indicate an **up** status. Multiple status codes should be separated by spaces. Specifying an asterisk (*****) indicates that all status codes are acceptable.

SMTP

An SMTP type of monitor checks the status of Simple Mail Transport Protocol (SMTP) servers. This monitor type is an extremely basic 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-type monitor requires a domain name.

Figure 10.25 shows the settings and default values of an SMTP-type monitor.

```
Name my_smtp
Type SMTP
Interval 5
Timeout 16
Domain ""
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.25 *An SMTP-type custom monitor with default values*

SNMP DCA and SNMP DCA Base

There are two types of SNMP performance monitors that you can configure: SNMP DCA and SNMP DCA Base. The following two sections describe these monitors, followed by the procedure for how to configure them using the Configuration utility.

SNMP DCA

With an SNMP DCA type of monitor, you can check the performance of a server running an SNMP agent such as UC Davis, for the purpose of load balancing traffic to that server. With this monitor you can define ratio weights for CPU, memory, and disk use.

Performance monitors are generally used with dynamic ratio load balancing. For more information on performance monitors and dynamic ratio load balancing, see Chapter 4, *Configuring Load Balancing Pools*, and Appendix A, *Additional Monitor Considerations*.

Note

Unlike health monitors, performance monitors do not report on the status of a pool, pool member, or node.

The LTM system provides a pre-configured SNMP DCA monitor named **snmp_dca**. Figure 10.26 shows the settings and values of the **snmp_dca** pre-configured monitor.

```
Name snmp_dca
Type SNMP DCA
Interval 10
Timeout 30
Community Public
Version v1
Agent Type UCD
CPU Coefficient 1.5
CPU Threshold 80
Memory coefficient 1.0
Memory Threshold 70
Disk Coefficient 2.0
Disk Threshold 90
Variables ""
Alias Address * All Addresses
Alias Service Port SNMP
```

***Figure 10.26** The **snmp_dca** pre-configured monitor*

Pre-configured monitors are not user-modifiable. Thus, if you want to change the values for the SNMP DCA monitor settings, you must create an SNMP DCA-type custom monitor. Possible values for the **Version** setting are **v1**, **v2c**, and **Other**. Possible values for the **Agent Type** setting are **UCD**, **Win2000**, and **Other**.

When configuring an SNMP DCA custom monitor, you can use the default CPU, memory, and disk coefficient and threshold values specified in the monitors, 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, such as **Win2000**.

SNMP DCA Base

You use an SNMP DCA Base type of monitor to check the performance of servers that are running an SNMP agent, such as UC Davis. However, you should use this monitor only when you want the load balancing destination to be based solely on user data, and not CPU, memory or disk use.

Figure 10.27 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.

```
Name my_snmp_dca_base
Type snmp_dca_base
Interval 10
Timeout 30
Community Public
Version v1
USEROID ".1.3.6.1.4.1.2021.4.6.0"
USEROID_COEFFICIENT "1.0"
USEROID_THRESHOLD "90"
Alias Address * All Addresses
Alias Service Port SNMP
```

Figure 10.27 *An SNMP DCA-type custom monitor with default values*

◆ 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 use the Configuration utility.

To configure an SNMP monitor using the Configuration utility

1. From the Main tab on the navigation pane, expand **Local Traffic**, and click **Monitors**.
   This displays a list of existing pre-configured and custom monitors.
2. Click **Create**.
   This displays the New Monitor screen.
3. In the **Name** box, type a unique name for the monitor.
4. From the **Type** box, select a monitor type:
   - If you want the monitor to include CPU, memory, disk, and user metrics, select **SNMP DCA**.
   - If you want the monitor to include user metrics only, select **SNMP DCA Base**.

5. From the **Import Settings** list, select the name of a pre-configured or custom SNMP-type monitor.
6. Retain or change the value in the **Interval** box.
   The default is 5 seconds. You can change this to any number you want.
7. Retain or change the value in the **Timeout** box.
   The default is 16 seconds. You can change this number to any number you want, however, it should be 3 times the interval number of seconds plus 1 second.
8. In the **Community** box, type a community name.
   The default value is **public**.
9. In the Version box, select the appropriate SNMP version number (**v1** or **v2c**).
10. If configuring an SNMP DCA monitor:
    a) Retain or change the values for CPU , memory, and disk coefficients and thresholds.
    b) Configure the **Variables** setting by specifying a unique name and a value for each **Name/Value** pair and clicking **Add**.
    The variables represent values for coefficient and threshold values for other types of data, such as user metrics.
11. If configuring an SNMP DCA Base monitor, configure the **Variables** setting by specifying a unique name and a value for each **Name/Value** pair and clicking **Add**.

    Note: *These variables correspond to USEROID, USEROID_COEFFICIENT, and USEROID_THRESHOLD values. Note that if the value of the USEROID variable is an absolute value, verify that the USEROID_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.*
12. For the **Alias Address** setting, type a destination IP address or retain the default value.
13. For the **Alias Service Port** setting, type a service port or retain the default value.
14. Click **Finished**.

Performance monitors are generally used with dynamic ratio load balancing. For more information on performance monitors and dynamic ratio load balancing, see Chapter 4, *Configuring Load Balancing Pools* and Appendix A, *Additional Monitor Considerations*.

◆ Note

Unlike health monitors, performance monitors do not report on the status of pool, pool member, or node.

SOAP

A SOAP monitor tests a Web service based on the Simple Object Access protocol (SOAP). More specifically, the monitor submits a request to a SOAP-based Web service, and optionally, verifies a return value or fault. Figure 10.28 shows the settings and default values of a SOAP-type monitor.

```
Name my_soap
Type soap
User Name ""
Password ""
Protocol" HTTP
URL Path ""
Namespace ""
Method ""
Parameter Name ""
Parameter Type bool
Parameter Value ""
Return Type bool
Return Value ""
Expect Fault No
Alias Address * All Addresses
Alias Service Port * All Ports
```

***Figure 10.28** A SOAP-type custom monitor with default values*

Possible values for the Protocol setting are **HTTP** and **HTTPS**.

Possible values for the **Parameter Type** setting are: **bool**, **int**, **long**, and **string**.

Possible values for the **Return Type** setting are: **bool**, **int**, **short**, **long**, **float**, **double**, and **string**.

Possible values for the **Expect Fault** setting are **No** and **Yes**.

UDP

You use a UDP type of monitor when the system is sending User Datagram Protocol (UDP) packets. Designed to check the status of a UDP service, a UDP-type monitor sends one or more UDP packets to a target pool, pool member, or node.

Figure 10.29 shows the settings and default values of a UDP-type monitor. As shown in this figure, the value in seconds of the **Timeout Packets** setting should be lower than the value of the **Interval** setting**.**

```
Name my_udp
Type UDP
Interval 5
Timeout 16
Send String default send string
Send Packets 2
Timeout Packets 2
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.29 *A UDP-type custom monitor with default values*

Important

When specifying a ***Send String*** *value, do not use the* ***Enter*** *key to specify a new line when more than one line is required for the string. Instead, type the escape characters* ***\n****.*

When using a UDP-type monitor to monitor a pool, pool member, or node, you must also enable another monitor type, such as **ICMP**, to monitor the pool, pool member, or node. Until both a UDP-type monitor and another type of monitor to report the status of the UDP service as **up**, the UDP service receives no traffic. See Table 10.4 for details.

If a UDP monitor reports status as	And another monitor reports status as	Then the UDP service is
up	up	up
up	down	down
down	up	down
down	down	down

Table 10.4 *Determining status of the UDP service*

WAP

You use a WAP monitor to monitor Wireless Application Protocol (WAP) servers. The common usage for the WAP monitor is to specify the **Send String** and **Receive String** settings only. The WAP monitor functions by requesting a URL (the **Send String** setting) and finding the string in the **Receive String** setting somewhere in the data returned by the URL response. Figure 10.30 shows the settings and default values of a WAP-type monitor.

```
Name my_wap
Type WAP
Import Settings wap
Interval 10
Timeout 31
Send String ""
Receive String ""
Secret ""
Accounting Node ""
Accounting Port ""
Server ID ""
Call ID ""
Session ID ""
Framed Address ""
Alias Address * All Addresses
Alias Service Port * All Ports
```

Figure 10.30 *A WAP-type custom monitor with default values*

Important

*When specifying **Send String** and **Receive String** values, do not use the **Enter** key to specify a new line when more than one line is required for the string. Instead, type the escape characters **\n**.*

The **Secret** setting is the RADIUS secret, a string known to both the client and the RADIUS server, and is used in computing the MD5 hash.

The **Accounting Node** setting specifies the RADIUS node. If this a null string and RADIUS accounting has been requested (accounting port is non-zero), then the WAP server node is assumed to also be the RADIUS node.

If set to non-zero, the **Accounting Port** setting requests RADIUS accounting and uses the specified port.

The **Server ID** setting specifies the RADIUS NAS-ID of the requesting server (that is, the BIG-IP system). It is a string used as an alias for the FQDN. See the section on testing WAP_monitor just below.

The **Call ID** setting is an identifier similar to a telephone number, that is, a string of numeric characters. For testing purposes, this value is usually a string of eleven characters.

The **Session ID** setting is a RADIUS session ID, used to identify this session. This is an arbitrary numeric character string, often something like **01234567**.

The **Framed Address** setting is a RADIUS framed IP address. The setting has no special use and is usually specified simply as **1.1.1.1**.

RADIUS accounting is optional. To implement RADIUS accounting, you must set the accounting port to a non-zero value. If you set the **Accounting Port** setting to a non-zero value, then the monitor assumes that RADIUS accounting is needed, and an accounting request is sent to the specified accounting node and port to start accounting. This is done before the URL is requested. After the successful retrieval of the URL with the correct data, an accounting request is sent to stop accounting.

WMI

A WMI type of monitor checks the performance of a pool, pool member, or node that is running the Windows Management Infrastructure (WMI) data collection agent and then dynamically load balances traffic accordingly.

You generally use performance monitors such as a WMI monitor with dynamic ratio load balancing. For more information on performance monitors and dynamic ratio load balancing, see Chapter 4, *Configuring Load Balancing Pools*, and Appendix A, *Additional Monitor Considerations*.

◆ Note

Unlike health monitors, performance monitors do not report on the status of a pool, pool member, or node.

Figure 10.31 shows the settings and default values of a WMI-type monitor.

```
Name my_wmi
Type wmi
Interval 5
Timeout 16
User Name ""
Password ""
Method POST
URL /scripts/f5Isapi.dll
Command GetCPUInfo, GetDiskInfo, GetOSInfo
Metrics LoadPercentage, DiskUsage, PhysicalMemoryUsage:1.5,
  VirtualMemoryUsage:2.0
Agent Mozilla/4.0 (compatible: MSIE 5.0; Windows NT)
Post RespFormat=HTML
Alias Address * All Addresses
Alias Service Port HTTP
```

***Figure 10.31** A WMI-type custom monitor with default values*

Note that when creating a custom WMI monitor, the only default values that you are required to change are the null values for user name and password. Also note that you cannot change the value of the **Method** setting.

Table 10.5 shows the complete set of commands and metrics that you can specify with the **Command** and **Metrics** settings. Also shown are the default metric values.

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
	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

Table 10.5 *WMI-type monitor commands and metrics*

Command	Metric	Default Coefficient	Default Threshold
	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
	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 10.5 *WMI-type monitor commands and metrics*

Special configuration considerations

Every pre-configured or custom monitor has settings with some default values assigned. The following sections contain information that is useful when changing these default values.

Setting destinations

By default, the value for the **Alias Address** setting in the monitors is set to the wildcard *** Addresses**, and the **Alias Service Port** setting is set to the wildcard *** Ports**. This value causes the monitor instance created for a pool, pool member, or 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 destination value, by creating a custom monitor. An explicit value for the **Alias Address** and/or **Alias Service Port** setting is used to force the instance destination to a specific address and/or port which may not be that of the pool, pool member, or node.

The ECV monitors **http, https,** and **tcp** have the settings **Send String** and **Receive String** for the send string and receive expression, respectively.

The most common **Send String** value is **GET /**, which retrieves a default HTML page for a web site. To retrieve a specific page from a web site, you can enter a **Send String** value that is a fully qualified path name:

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

The **Receive String** expression is the text string the monitor looks for in the returned resource. The most common **Receive String** 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 **Receive String** value ["**""**]. In this case, any content retrieved is considered a match. If both the **Send String** and **Receive String** are left empty, only a simple connection check is performed.

For HTTP and FTP monitors, you can use the special settings **get** or **hurl** in place of **Send String** and **Receive String** statements. For FTP monitors specifically, the **GET** setting specifies the full path to the file to retrieve.

Using transparent and reverse modes

The normal and default behavior for a monitor is to ping the destination pool, pool member, or node by an unspecified route, and to mark the node **up** if the test is successful. However, with certain monitor types, you can specify a route through which the monitor pings the destination server. You configure this by specifying the **Transparent** or **Reverse** setting within a custom monitor.

- **Transparent setting**
  Sometimes it is necessary to ping the aliased destination through a transparent pool, pool member, or node. When you create a custom monitor and set the **Transparent** setting to **Yes**, the LTM system forces the monitor to ping *through* the pool, pool member, or node with which it is associated (usually a firewall) to the pool, pool member, or node. (In other words, if there are two firewalls in a load balancing pool, the destination pool, pool member, or node is always pinged through the pool, pool member, or node specified and not through the pool, pool member, or node selected by the load balancing method.) In this way, the transparent pool, pool member, or node is tested: if there is no response, the transparent pool, pool member, or 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 create a monitor called **http_trans** in which you specify **10.10.10.53:80** as the monitor destination address, and set the **Transparent** setting to **Yes**. Then you associate the monitor **http_trans** with the transparent pool, pool member, or node.

  This causes the monitor to check the address **10.10.10 53:80** through **10.10.10.101:80**. (In other words, the LTM system routes the check of **10.10.10.53:80** 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 pool members or nodes, see *Associating monitors with pools and nodes*, on page 10-39.

- **Reverse setting**
  With the **Reverse** setting set to **Yes**, the monitor marks the pool, pool member, or 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**.

Figure 10.6 shows the monitors that contain the **Transparent** setting, the **Reverse** setting, or both.

Monitor Type	Setting	
TCP	Transparent	Reverse
HTTP	Transparent	Reverse
HTTP		Reverse
TCP Echo	Transparent	
ICMP	Transparent	

Table 10.6 *Monitors that contain the* ***Transparent*** *or* ***Reverse*** *settings*

Associating monitors with pools and nodes

Once you have created a monitor and configured its settings, the final task is to associate the monitor with the server or servers to be monitored. The server or servers can be either a pool, a pool member, or a node, depending on the monitor type.

Some monitor types are designed for association with nodes only, and not pools or pool members. Other monitor types are intended for association with pools and pool members only, and not nodes. Therefore, when you use the Configuration utility to associate a monitor with a pool, pool member, or node, the utility displays only those pre-configured monitors that are designed for association with that server. For example, you cannot associate the monitor **icmp** with a pool or its members, since the **icmp** monitor is designed to check the status of a node itself and not any service running on that node.

When you associate a monitor with a server, the LTM system automatically creates an ***instance*** of that monitor for that server. A monitor association thus creates an instance of a monitor for each server that you specify. Therefore, you can have multiple instances of the same monitor running on your servers.

The Configuration utility allows you to disable an instance of a monitor that is running on a server. This allows you to suspend health or performance checking, without having to actually remove the monitor association. When you are ready to begin monitoring that server again, you simply re-enable that instance of the monitor.

Types of monitor associations

The three types of monitor associations are:

- **Monitor-to-pool association**
  This type of association associates a monitor with an entire load balancing pool. In this case, the monitor checks all members of the pool. For example, you can create an instance of the monitor **http** for every member of the pool **my_pool**, thus ensuring that all members of that pool are checked.
- **Monitor-to-pool member association**
  This type of association associates a monitor with an individual pool member, that is, an IP address and service. In this case, the monitor checks only that pool member and not any other members of the pool. For example, you can create an instance of the monitor **http** for pool member **10.10.10.10:80** of **my_pool**.
- **Monitor-to-node association**
  This type of association associates a monitor with a specific node. In this case, the monitor checks only the node itself, and not any services running on that node. For example, you can create an instance of the monitor **icmp** for node **10.10.10.10**. In this case, the monitor checks the specific node only, and not any services running on that node.

For more information on associating monitors with pools and pool members, see Chapter 4, *Configuring Load Balancing Pools*. For more information on associating monitors with nodes, see Chapter 3, *Configuring Nodes*.

Managing monitors

When managing existing monitors, you can display or delete them, or you can enable and disable an instance of a monitor. Note that prior to deleting a monitor, you must remove all existing monitor associations.

To display a monitor

1. On the Main tab, expand **Local Traffic**.
2. Click **Monitors.**
   A list of existing monitors appears.
3. Click a monitor name.
   This displays the monitor settings and their values.

To delete a monitor

1. On the Main tab, expand **Local Traffic**.
2. Click **Monitors.**
   A list of existing monitors appears.
3. Click the Select box for the monitor that you want to delete.
4. Click **Delete**.
   A confirmation message appears.
5. Click **Delete**.

To enable or disable a monitor instance

1. On the Main tab, expand **Local Traffic**.
2. Click **Monitors**.
   This displays a list of custom monitors.
3. Click a monitor name in the list.
4. On the menu bar, click **Instances**.
   This lists any existing monitor instances.
5. For the instance you want to manage, click the **Select** box.
6. Click **Enable** or **Disable**.
7. Click **Update**.

11

Configuring SNATs and NATs

- Introducing secure network address translation
- Creating a SNAT pool
- Implementing a SNAT
- Implementing a NAT
- Managing SNATs and NATs
- SNAT examples

Introducing secure network address translation

A virtual server configured on a BIG-IP® local traffic management (LTM) system translates the destination IP address of an incoming packet to another destination IP address, for the purpose of load balancing that packet. Normally, the source IP address remains unchanged.

As an option, you can also create a secure network address translation (SNAT). A ***SNAT*** is an object that maps an original client IP address (that is, a source IP address) to a translation address that you choose. Thus, a SNAT causes the LTM system to translate the source IP address of an incoming packet to an address that you specify. The purpose of a SNAT is simple: to ensure that the target server sends its response back through the LTM system rather than to the original client IP address directly.

To create a SNAT, you either use the Configuration utility or write an iRule, depending on the type of SNAT you are creating.

◆ Note

This type of translation has no effect on the destination address translation that a virtual server performs.

Examples of scenarios where a SNAT is useful are:

- To connect to an external device that requires a routable return IP address
- To connect to a virtual server with a node that is on the same IP subnet as the client

◆ Tip

Because the purpose of a SNAT is simply to change the source IP address of incoming packets, the term ***secure network address translation*** *is a slight misnomer. A better way to define the SNAT acronym would be* ***source network address translation****, or* ***source NAT****.*

How does a SNAT work?

A SNAT works in the following way:

1. The LTM system receives a packet from an original client IP address and checks to see if that source address is defined in a SNAT.
2. If the client's IP address is defined in a SNAT, the LTM system changes that source IP address to the translation address defined in the SNAT.
3. The LTM system then sends the client request, with the SNAT translation address as the source address, to the target server.

The end result of this process is that the target server has a routable IP address for the client that the server can specify as the destination IP address in its response.

Mapping original IP addresses to translation addresses

When you create a SNAT, you map an original IP address to a translation address in one of several ways, depending on your needs. For example, you can explicitly map an original IP address to a single translation address, or you can create a *pool* of translation addresses and map the original IP address to that pool of addresses.

Mapping a specific original IP address to a specific translation address

One way to create a SNAT is to directly map one or more original IP addresses to a specific translation address that you choose. A SNAT that you create in this way is a type of standard SNAT. A ***standard SNAT*** is a SNAT object that you create using the New SNAT screen of the Configuration utility. For more information on standard SNATs, see *Implementing a SNAT*, on page 11-6.

Using the SNAT automap feature

Another way to create a SNAT is to use a feature of the LTM system called SNAT automap. The ***SNAT automap*** feature automatically maps one of the system's self IP addresses to the original IP address you specify during SNAT creation. When you use this feature, you do not need to explicitly specify a translation address.

A SNAT that you create in this way is a type of standard SNAT. For more information on standard SNATs, see *Implementing a SNAT*, on page 11-6.

Mapping a specific original IP address to a pool of translation addresses

You can also create a SNAT by creating a pool of translation addresses and then mapping an original IP address to the entire translation pool. This pool of translation addresses is known as a ***SNAT pool***. You create a SNAT pool using the New SNAT Pool screen of the Configuration utility. For information on creating a SNAT pool, see *Implementing a SNAT*, on page 11-6.

Once you have created a SNAT pool and mapped it to an original IP address, and the virtual server then receives a packet from the original IP address, the LTM system chooses a translation address from that SNAT pool. The system then translates the original IP address to the chosen address.

You can map an original IP address to the SNAT pool in one of two ways:

- **By creating a SNAT object**.
  A SNAT that you create this way, using the New SNAT screen in the Configuration utility, is a type of standard SNAT. For more information on standard SNATs, see *Creating a standard SNAT*, on page 11-6.
- **By writing an iRule**.
  In this case, you do not create a SNAT object. Instead, you write an iRule that includes a **snat** or **snatpool** command. The type of SNAT that you create by writing an iRule is called an intelligent SNAT. An ***intelligent SNAT*** is the mapping of one or more original client IP addresses to a translation address through the use of an iRule. For more information on intelligent SNATs, see *Creating an intelligent SNAT*, on page 11-9.

Mapping all original IP addresses to a pool of translation addresses

Yet another way to create a SNAT is to create a SNAT pool (using the New SNAT Pool screen of the Configuration utility) and directly assign it to a virtual server as a resource of that virtual server. Once you have assigned a SNAT pool to a virtual server, the LTM system automatically maps all original IP addresses coming through the virtual server to that SNAT pool. As with intelligent SNATs, you do not create a SNAT object, with the New SNAT screen, in the Configuration utility. For more information on this type of SNAT, see *Assigning a SNAT pool directly to a virtual server*, on page 11-10.

Creating a SNAT pool

If you decide that you want to use a SNAT pool as the way to specify translation addresses in your SNAT, you must first create the SNAT pool, specifying one or more translation addresses that you want to include in the SNAT pool. You create a SNAT pool using the Configuration utility. For background information on SNAT pools, see *Mapping a specific original IP address to a pool of translation addresses*, on page 11-3.

After creating the SNAT pool, you then create the type of SNAT that best suits your needs (a standard SNAT, an intelligent SNAT, or a SNAT pool that you assign directly to a virtual server). To understand the different types of SNATs that you can create, see *Implementing a SNAT*, on page 11-6.

A SNAT pool has two settings that you must configure when you create it. Table 11.1 lists and describes these settings.

Property	Description	Default Value
Name	The unique name of the SNAT pool.	No default value
Member List	The list of IP addresses that you want to include in SNAT pool. If the IP addresses that you add are not already designated as translation addresses, the LTM system automatically designates them as such and assigns them the appropriate properties with their default values. This setting is required.	No default value

Table 11.1 *Properties of a SNAT pool*

Each translation address that you add to the SNAT pool has settings that you can configure after you add the address to the SNAT pool. For information on these settings, see *Specifying a translation address*, on page 11-8.

Once you create a SNAT pool, you must do one of the following:

- Reference the SNAT pool from within a SNAT object that you create. You do this when you create a standard SNAT. For more information, see *Creating a standard SNAT*, on page 11-6.
- Reference the SNAT pool from within an iRule and then assign the iRule to a virtual server as a resource. You do this when you create an intelligent SNAT. For more information, see *Creating an intelligent SNAT*, on page 11-9.
- Assign the SNAT pool directly to a virtual server as a resource. For more information, see *Assigning a SNAT pool directly to a virtual server*, on page 11-10.

To create a SNAT pool

1. On the Main tab, expand **Local Traffic**.
2. Click **SNATs**.
   This displays a list of existing SNATs.
3. Click **SNAT Pool List** on the menu bar.
   This displays a list of existing SNAT pools.
4. In the upper-right corner of the screen, click **Create**.
5. For the **Name** setting, type a unique name for the SNAT pool.
6. For the **Member List** setting, type an IP address.
7. Click **Add**.
8. Repeat steps 6 and 7 for each translation address that you want to add.
9. Click **Finished**.

Implementing a SNAT

Before implementing secure network address translation, you should decide which type of SNAT you want to create. The types of SNATs you can create are:

- **Standard SNAT**
  A ***standard SNAT*** is an object you create, using the Configuration utility, that specifies the mapping of one or more original client IP addresses to a translation address. For this type of SNAT, the criteria that the LTM system uses to decide when to apply the translation address is based strictly on the original IP address. That is, if a packet arrives from the original IP address that you specified in the SNAT, then the LTM system translates that address to the specified translation address.

  There are three types of standard SNATs that you can create:

  - A SNAT in which you specify a specific translation address
  - A SNAT that uses the automap feature
  - A SNAT in which you specify a SNAT pool as your translation address

- **Intelligent SNAT**
  Like a standard SNAT, an ***intelligent SNAT*** is the mapping of one or more original client IP addresses to a translation address. However, you implement this type of SNAT mapping within an iRule instead of by creating a SNAT object. For this type of SNAT, the criteria that the LTM system uses to decide when to apply a translation address is based on any piece of data you specify within the iRule, such as an HTTP cookie or a server port.

- **SNAT pool assigned as a virtual server resource**
  This type of SNAT consists of just a SNAT pool that you directly assign as a resource to a virtual server. When you implement this type of SNAT, you create a SNAT pool only; you do not need to create a SNAT object or an iRule.

For more information on mapping original IP addresses to translation addresses, see *Mapping original IP addresses to translation addresses*, on page 11-2.

Creating a standard SNAT

You create a standard SNAT using the Configuration utility. The translation address or addresses that you map to an original IP address can be either a specific IP address, an existing SNAT pool, or a self IP address (using the automap feature).

When you create a standard SNAT, the LTM system automatically assigns a set of properties to the SNAT. While you must configure the **Name** and **Translation** settings at the time that you create the SNAT, you can use the default values for the other settings, or modify those values later.

To create a standard SNAT

1. On the Main tab, expand **Local Traffic**.
2. Click **SNATs**.
   This displays a list of existing SNATs.
3. In the upper-right corner of the screen, click **Create**.
4. For the **Name** setting, type a unique name for the SNAT.
5. For the **Translation** setting, select **IP Address**, **SNAT Pool**, or **Automap**.
6. If you selected **IP Address** or **SNAT Pool**, type an IP address or select a SNAT pool name.
7. Change or retain all other values.
8. Click **Finished**.

Table 11.2 shows the settings that you can configure for a SNAT. Following the table are detailed descriptions of each setting.

Property	Description	Default Value
Name	Specifies the unique name of the standard SNAT. Setting this property is required.	No default value
Translation	Depending on the value selected, specifies an individual IP address, a SNAT pool name, or the Automap option. Possible values are: **IP Address**, **SNAT Pool**, or **Automap**.	**Automap**
Origin	Specifies the original client IP addresses to which you want to map a translation address or pool of translation or self IP addresses. Possible values are **All Addresses** or **Address List**.	**All Addresses**
VLAN Traffic	The VLAN to which you want the SNAT to apply. Possible values are: **ALL VLANS**, **Enabled On**, and **Disabled On**.	**ALL VLANS**

Table 11.2 *Properties of a standard SNAT*

Specifying a SNAT name

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

Each SNAT that you define must have a unique name.

Specifying a translation address

The **Translation** setting specifies the translation addresses that you want to map to your original client IP addresses. For background information on translation addresses, see *Mapping original IP addresses to translation addresses*, on page 11-2.

There are three possible values for the **Translation** setting:

- **IP Address**
  When creating a SNAT, you can specify a particular IP address that you want the SNAT to use as a translation address. For the procedure on specifying a particular translation address, see *To explicitly define a translation address*, on page 11-14.
- **SNAT pool**
  Specifying this value allows you to specify an existing SNAT pool to which you want to map your original client IP address. For information on SNAT pools and how to create them, see *Creating a SNAT pool*, on page 11-4. For an example of a standard SNAT that uses a SNAT pool, see *Example 1 - Establishing a standard SNAT that uses a SNAT pool*, on page 11-17.
- **Automap**
  Similar to a SNAT pool, the SNAT automap feature allows you to map one or more original client IP addresses to a pool of translation addresses. However, with the SNAT automap feature, you do not need to create the pool. Instead, the LTM system effectively creates a pool for you, using all of the LTM system's self IP addresses as the translation addresses for the pool.

When you specify a translation address or a SNAT pool, the LTM system automatically assigns a set of properties to that translation address. You can use the default values for these properties, or you can change them to suit your needs. Table 11.3 lists and describes the properties of a translation address.

Property	Description	Default Value
IP address	The IP address that you want to designate as a translation address. This is a required setting.	No default value
State	The state of the translation address, that is, enabled or disabled. If set to **disabled**, the translation address is not used to initiate a connection.	**Enabled**
ARP	A setting that determines whether or not the LTM system responds to ARP requests or sends gratuitous ARPs.	**Enabled**
Connection Limit	A limit on the number of connections a translation address must reach before it no longer initiates a connection. The default value of **0** indicates that the setting is disabled.	**0**

Table 11.3 *Properties of a SNAT translation address*

Property	Description	Default Value
TCP Idle Timeout	A timer that defines the number of seconds that TCP connections initiated using a SNAT address are allowed to remain idle before being automatically disconnected. Possible values are **Indefinite** or **Specify**.	**Indefinite**
UDP Idle Timeout	A timer that defines the number of seconds that UDP connections initiated using a SNAT address are allowed to remain idle before being automatically disconnected. Possible values are **Indefinite** or **Specify**.	**Indefinite**
IP Idle Timeout	A timer that defines the number of seconds that IP connections initiated using a SNAT address are allowed to remain idle before being automatically disconnected. Possible values are **Indefinite** or **Specify**.	**Indefinite**

Table 11.3 *Properties of a SNAT translation address*

Specifying original IP addresses

The **Origin** setting specifies the original client IP addresses that you want to map to translation addresses. You can add one IP address or multiple IP addresses as values for this setting.

Specifying VLAN traffic

The **VLAN Traffic** setting specifies the VLANs to which you want the SNAT to apply. Possible values are: **ALL VLANS**, **Enabled On**, and **Disabled On**.

Creating an intelligent SNAT

One way to perform secure address translation is to create an intelligent SNAT. As described previously, an ***intelligent SNAT*** is not a SNAT object, but instead an iRule that maps of one or more original client IP addresses to a translation address. To create an intelligent SNAT, you must complete these tasks:

- If you are mapping an original IP address to a SNAT pool (as opposed to an individual translation address), use the New SNAT Pools screen to create one or more SNAT pools that include those translation addresses as members. For more information, see *To create a SNAT pool*, on page 11-5.
- Use the New Rules screen to create an iRule that includes the **snat** or **snatpool** command. These iRule commands specify the translation address or the pool of translation addresses that the LTM system should use to select a translation address. For more information on iRules™, see Chapter 13, *Writing iRules*.

- From the Resources screen for the appropriate virtual server, assign the iRule as a resource to the virtual server. For more information on virtual servers, see Chapter 2, *Configuring Virtual Servers*.

Note

For an example of an intelligent SNAT, see ***Example 2 - Establishing an intelligent SNAT****, on page 11-18.*

Assigning a SNAT pool directly to a virtual server

Rather than creating a SNAT object, or an intelligent SNAT using an iRule, you have the option of simply creating a SNAT pool and then assigning it as a resource directly to a virtual server. This eliminates the need for you to explicitly define original IP addresses to which to map translation addresses.

Implementing a NAT

A ***network translation address*** (NAT) provides an 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 initiate but not receive a connection.)

The IP addresses that identify nodes on the 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 external servers, such as mail servers or databases.

Using NATs solves this problem. NATs assign to a particular node a routable IP address that the node can use as its source IP address when connecting to external servers. You can use the NAT IP address to connect directly to the node through the LTM system, rather than having the LTM system send the traffic to a random node according to the specified load balancing method.

◆ 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 must create a separate NAT for each node, using the Configuration utility. When you create a NAT, you configure a set of properties. While you must configure the **NAT Address** and **Origin Address** settings at the time that you create the NAT, you can use the default values for the other settings, or modify those values later.

To create a NAT

1. On the Main tab, expand **Local Traffic**.
2. Click **SNATs**.
   The SNATs screen opens.
3. Click the **NAT List** menu.
4. In the upper right corner, click **Create**.
   The New NAT screen opens.
5. In the **NAT Address** box, type the IP address that you want to use as a translation address.
6. In the **Origin Address** box, type the original client IP address to be translated.
7. Retain or modify all other values as necessary.
8. Click **Finished**.

Table 11.4 shows the settings that you can configure for a NAT, with a description of each.

NAT Attribute	Description	Default Value
NAT Address	An IP address that is routable on the external network of the LTM system.	No default value
Origin Address	The original address is the node IP address of a host that you want to be able to connect to through the NAT.	No default value
State	The state of the NAT, that is, whether the NAT is enabled or disabled.	**Enabled**
ARP	A setting that instructs the LTM system to respond to ARP requests from the specified NAT address, and send gratuitous ARP requests for router table updates.	**Enabled**
VLAN Traffic	VLANs to which the NAT is not to be mapped can be explicitly disabled, as when there is more than one internal VLAN.	**All VLANS**

Table 11.4 *NAT configuration settings*

In addition to these options, you can set up forwarding virtual servers that allow you to selectively forward traffic to specific addresses.

Additional restrictions

When using a NAT, you should be aware of the following restrictions:

- The IP address defined in the **Origin Address** box must be routable to a specific server behind the system.
- You must delete a NAT before you can redefine it.

Managing SNATs and NATs

Using the Configuration utility, you can manage existing SNATs in many ways. For example, you might want to view a list of existing SNAT pools before creating a new one. Or you might want to modify the way that a standard SNAT maps an original IP address to a translation address.

That tasks that you can perform when managing SNATs are:

- Viewing or modify a SNAT or NAT, or a SNAT pool
- Defining or viewing translation addresses
- Deleting SNATs or NATs, SNAT pools, and translation addresses
- Enabling or disabling SNATs or NATs for a load balancing pool
- Enabling or disabling SNAT or NAT translation addresses

Viewing or modifying SNATs, NATs, and SNAT pools

You can view or modify any SNATs, NATS, or SNAT pools that you created previously.

To view or modify a SNAT or NAT

1. On the Main tab, expand **Local Traffic**.
2. Click **SNATs**.
   This displays a list of existing SNATs.
3. Select the type of item you want to view:
   - If you want to view or modify a SNAT, click a SNAT name.
   - If you want to view or modify a NAT, find the **NAT List** menu, and click a NAT address.
4. View or modify the displayed settings.
5. If you modified any settings, click **Update**.

To view or modify a SNAT pool

1. On the Main tab, expand **Local Traffic**.
2. Click **SNATs**.
   This displays a list of existing SNATs.
3. On the menu bar, click the SNAT Pool List menu.
   This displays a list of existing SNAT pools.
4. Click a SNAT pool name.
5. View or modify the displayed settings.
6. If you modified any settings, click **Update**.

Defining and viewing translation addresses

You can define a translation address or view any existing translation addresses the you defined previously.

To explicitly define a translation address

1. On the Main tab, expand **Local Traffic**.
2. Click **SNATs**.
3. On the menu bar, click the **SNAT Translation List** menu.
   This displays any existing translation addresses.
4. In the upper-right corner of the screen, click **Create**.
5. Retain or change all property settings.
6. Click **Finished**.

To view translation addresses

1. On the Main tab, expand **Local Traffic**.
2. Click **SNATs**.
   This displays a list of existing SNATs.
3. On the menu bar, click the **SNAT Translation List** menu.
   This displays a list of existing translation addresses.
4. Click a translation address.
5. View or modify the displayed settings.
6. If you modified any settings, click **Update**.

Deleting SNATs, NATs, SNAT pools, and translation addresses

You can delete any existing SNAT, NAT, SNAT pool, or translation address that you created previously.

◆ Note

When you delete a SNAT, the BIG-IP system only deletes the SNAT if no connection is actively using it. To delete SNATs that are still in use, you must issue the ***bigstart restart*** *command.*

To delete a SNAT or a NAT

1. On the Main tab, expand **Local Traffic**.
2. Click **SNATs**.
   This displays a list of existing SNATs.
3. Select the type of item you want to delete:

- If you want to delete a SNAT, locate the SNAT you want to delete, and check the Select box on the left.
- If you want to delete a NAT, click the **NAT List** menu, locate the NAT you want to delete, and check the Select box to the left.

4. At the bottom of the screen, click **Delete**.

To delete a SNAT pool

1. On the Main tab, expand **Local Traffic**.
2. Click **SNATs**.
   This displays a list of existing SNATs.
3. On the menu bar, click the **SNAT Pool List** menu.
   This displays a list of existing SNAT pools.
4. Locate the SNAT pool you want to delete, and check the Select box to the left.
5. At the bottom of the screen, click **Delete**.

To delete a translation address

1. On the Main tab, expand **Local Traffic**.
2. Click **SNATs**.
   This displays a list of existing SNATs.
3. On the menu bar, click the **SNAT Translation List** menu
   This displays a list of existing translation addresses.
4. Locate the translation address you want to delete, and check the Select box to the left.
5. At the bottom of the screen, click **Delete**.

Enabling or disabling SNATs or NATs for a load balancing pool

When configuring a load balancing pool, you can specifically disable SNAT or NAT translations on any connections that use that pool. By default, this setting is enabled. For more information, see Chapter 4, *Configuring Load Balancing Pools*.

Enabling or disabling SNAT translation addresses

Using the Configuration utility, you can enable or disable an individual SNAT translation address.

To enable or disable a SNAT translation address

1. On the Main tab, expand **Local Traffic**.

2. Click **SNATs**.
3. On the menu bar, click the **SNAT Translation List** menu.
4. Locate the translation address you want to enable or disable, and check the Select the box to the left.
5. At the bottom of the screen, click **Enable** or **Disabl**e.

SNAT examples

The following examples demonstrate ways to implement SNATs that make use of SNAT pools. The examples illustrate how you can:

- Establish a standard SNAT that uses a SNAT pool
- Establish an intelligent SNAT

Note

To best illustrate SNATs that use SNAT pools, the following examples show sample entries from the LTM system's ***bigip.conf*** *file. Entries in the* ***bigip.conf*** *file represent the result of using the Configuration utility to configure the LTM system.*

Example 1 - Establishing a standard SNAT that uses a SNAT pool

In some cases, you might need to create a SNAT that maps an original IP address to a SNAT pool instead of to an individual translation address. To illustrate this type of SNAT, suppose an ISP wants to provide two customers with two routable IP addresses each, for links to the Internet. The customers need to use these routable IP addresses as virtual IP addresses for inbound traffic to their own servers, and as translation addresses for outbound traffic from their servers.

In this case, the SNAT provides the solution. To implement the SNAT, the ISP takes the following three steps.

First, the ISP creates the load balancing pool **isp_pool**, shown in Figure 11.1.

```
pool isp_pool {
lb_method rr
member 199.5.6.254:0
member 207.8.9.254:0
}
```

Figure 11.1 *bigip.conf entries for a basic load balancing pool*

Next, the ISP creates three SNAT pools: **customer1_snatpool**, **customer2_snatpool**, and **other_snatpool**. This is shown in Figure 11.2. Note that the LTM system automatically designates the SNAT pool members as translation addresses.

```
snatpool customer1_snatpool {
member 199.5.6.10
member 207.8.9.10
}
snatpool customer2_snatpool {
member 199.5.6.20
member 207.8.9.20
}
snatpool other_snatpool {
member 199.5.6.30
member 207.8.9.30
}
```

Figure 11.2 *bigip.conf entries for three SNAT pools*

Finally, using the Configuration utility, the ISP creates a SNAT that maps each original IP address directly to the appropriate SNAT pool. Figure 11.3 shows these mappings as they appear in the **bigip.conf** file.

```
snat map {
192.1.1.10 192.1.1.11 to snatpool customer1_snatpool
}

snat map {
192.1.1.20 192.1.1.21 to snatpool customer2_snatpool
}

snat map default to snatpool other_snatpool
```

Figure 11.3 *bigip.conf entries that map original addresses to SNAT pools*

Example 2 - Establishing an intelligent SNAT

If you want to base SNAT mapping on criteria other than the original client IP address, such as a server port, you can write an iRule and specify a SNAT pool within the iRule. In this case, you use the SNAT screens in the Configuration utility to create a SNAT pool only, and not an actual SNAT object.

For example, suppose a user such as an ISP has two redundant connections to the Internet. In addition, the ISP handles many simultaneous CHAT connections (using port **531**), and wants to avoid exhausting the supply of server-side client ports. Finally, the ISP wants to collect statistics separately for CHAT, SMTP, and all other traffic. In this case, configuring an intelligent SNAT is the best way to choose the translation address.

To implement the intelligent SNAT, the ISP takes the following steps.

First, the ISP creates a load balancing pool called **out_pool**. In the **bigip.conf** file, the pool looks like the sample in Figure 11.4.

```
pool out_pool {
lb_method round_robin
member 199.5.6.254:0
member 207.8.9.254:0
}
```

Figure 11.4 *bigip.conf entries for a pool to be used in an intelligent SNAT*

Next, as shown in Figure 11.5, the ISP uses the Configuration utility to create a SNAT pool called **chat_snatpool** containing four IP addresses: **199.5.6.10**, **199.5.6.11**, **207.8.9.10**, and **207.8.9.11**. The LTM system automatically designates these IP addresses as translation addresses during creation of the SNAT pool. These addresses correspond to each of the two next hop networks that are to be used for CHAT traffic. In the **bigip.conf** file, the SNAT pool looks like the sample in Figure 11.5.

```
snatpool chat_snatpool {
member 199.5.6.10
member 199.5.6.11
member 207.8.9.10
member 207.8.9.11
}
```

Figure 11.5 *A SNAT pool definition for CHAT traffic*

Next, for each translation address, the ISP uses the Configuration utility to change the timeout value for TCP connections to **600**.

Then the ISP creates a second SNAT pool, **smtp_snatpool** containing two translation addresses: **199.5.6.20** and **207.8.9.20**. Each address corresponds to one of the two next hop networks that are to be used for SMTP traffic. In the **bigip.conf** file, the SNAT pool looks like the sample in Figure 11.6.

```
snatpool smtp_snatpool {
member 199.5.6.20
member 207.8.9.20
}
```

Figure 11.6 *A SNAT pool definition for SMTP traffic*

Next, the ISP creates the SNAT pool **other_snatpool** for all other traffic (that is, non-CHAT and non-SMTP traffic), where each IP address corresponds to one of the two next hop networks that are to be used by all other traffic. This is shown in Figure 11.7.

```
snatpool other_snatpool { \SNAT pool definition
member 199.5.6.30
member 207.8.9.30
}
```

Figure 11.7 *A SNAT pool definition for all other traffic*

Then the ISP writes an iRule that selects both a SNAT pool, based on the server port of the initiating packet, and the load balancing pool **out_pool**. Figure 11.9, on page 11-20, shows how the iRule specifies the command **TCP::local_port** to indicate the type of packet data to be used as a basis for selecting translation addresses. The iRule also shows the command **snatpool** (shown in figure 11.8) to specify the SNAT pools from which the LTM system is to select the translation addresses.

```
rule my_iRule {
when SERVER_CONNECTED
if ( TCP::local_port equals 531 ) {
use snatpool chat_snatpool
}
else if ( TCP::local_port equals 25 ) {
use snatpool smtp_snatpool
}
else {
use snatpool other_snatpool
}
use pool out_pool
}
```

Figure 11.8 *Example of an iRule that references an intelligent SNAT*

The **if** statement in the iRule instructs the LTM system to test the value of server port specified in the header of the client request. Based on the results, the LTM system selects both a SNAT pool and a load balancing pool.

As a final step, the ISP assigns the iRule as a resource to a wildcard virtual server, as shown in Figure 11.9.

```
virtual 0.0.0.0:0 use rule my_iRule
```

Figure 11.9 *Assignment of an iRule to a wildcard virtual server*

12

Configuring Rate Shaping

- Introducing rate shaping
- Creating and implementing rate classes
- Configuring rate class settings
- Managing rate classes

Introducing rate shaping

The BIG-IP® local traffic management (LTM) system includes a feature called rate shaping. ***Rate shaping*** allows you to enforce a throughput policy on incoming traffic. Throughput policies are useful for prioritizing and restricting bandwidth on selected traffic patterns.

Rate shaping can be useful for an e-commerce site that has preferred clients. For example, the site might want to offer higher throughput for preferred customers, and lower throughput for other site traffic.

The rate shaping feature works by first queuing selected packets under a rate class, and then dequeuing the packets at the indicated rate and in the indicated order specified by the rate class. A ***rate class*** is a rate-shaping policy that defines throughput limitations and a packet scheduling method to be applied to all traffic handled by the rate class.

You configure rate shaping by creating one or more rate classes and then assigning the rate class to a packet filter or to a virtual server. You can also use the iRules™ feature to instruct the LTM system to apply a rate class to a particular connection.

You can apply a rate class specifically to traffic from a server to a client or from a client to a server. If you configure the rate class for traffic that is going to a client, the LTM system does not apply the throughput policy to traffic destined for the server. Conversely, if you configure the rate class for traffic that is going to a server, the LTM system does not apply the throughput policy to traffic destined for the client.

To configure rate shaping, you use the Rate Shaping screens within the Local Traffic section of the Configuration utility.

Creating and implementing rate classes

A rate class defines the throughput limitations and packet scheduling method that you want the LTM system to apply to all traffic that the rate class handles. You assign rate classes to virtual servers and packet filter rules, as well as through iRules.

If the same traffic is subject to rate classes that you have assigned from more than one location, the LTM system applies the last-assigned rate class only. The LTM system applies rate classes in the following order:

- The first rate class that the LTM system assigns is from the last packet filter rule that matched the traffic and specified a rate class.
- The next rate class that the LTM system assigns is from the virtual server; if the virtual server specifies a rate class, the rate class overrides any rate class that the packet filter selects.
- The last rate class assigned is from the iRule; if the iRule specifies a rate class, this rate class overrides any previously-selected rate class.

To create a rate class

1. On the Main tab, expand **Local Traffic**.
2. Click **Rate Shaping**.
   This displays a list of existing rate classes.
3. In the upper-right corner of the screen, click **Create**.
   This displays the New Rate Class screen.
4. Specify whether you want to enable the rate class to borrow bandwidth from a parent rate class:
   - If you do not want the rate class to borrow bandwidth from a parent class, select **Basic**. For more information, see *Borrowing bandwidth*, on page 12-7.
   - If you want to enable the rate class to borrow bandwidth from a parent class, select **Advanced**. For more information, see *Specifying a parent class*, on page 12-7.
5. Configure all settings as needed.
   For information on settings, see *Configuring rate class settings*, on page 12-3, or see the online help.
6. Click **Finished**.

After you have created a rate class, you must assign it to a virtual server or a packet filter rule, or you must specify the rate class from within an iRule.

- For more information on virtual servers, see Chapter 2, *Configuring Virtual Servers*.
- For more information on packet filter rules, access the Packet Filters screens within the Configuration utility and display the online help.
- For more information on iRules, see Chapter 13, *Writing iRules*.

Configuring rate class settings

When you create a rate class, the LTM system assigns some default settings to the rate class. You can retain these default settings or modify them to suit your needs. The settings that you can configure for a rate class are described in Table 12.1.

Setting	Description	Default Value
Name	Specifies a unique name for the rate class. Every rate class requires a name.	No default value
Base Rate	Specifies the base throughput rate allowed for traffic that the rate class handles. Packets are generally not allowed to exceed the specified rate. This setting is required.	No default value
Ceiling Rate	Similar to the base rate, but specifies a hard, absolute limit. This number specifies the absolute limit on the rate at which traffic is allowed to flow when bursting or borrowing. For information on bandwidth bursting and borrowing, see *Specifying a burst size*, on page 12-4.	Same as Base Rate
Burst Size	Specifies the maximum number of bytes that traffic is allowed to burst beyond the base rate, before needing to borrow bandwidth. When this value is set to **0**, no bursting is allowed. For information on bandwidth bursting and borrowing, see *Specifying a burst size*, on page 12-4.	**0**
Direction	Specifies the direction of traffic to which the rate class is applied. Possible values are **Any**, **Client**, and **Server**.	**Any**
Parent Class	Specifies the rate class from which this class can borrow bandwidth. A child rate class can borrow any unused bandwidth from the parent rate class, thereby supplementing the burst size of the child rate class. This is an **Advanced** setting. For information on bandwidth bursting and borrowing, see *Specifying a burst size*, on page 12-4.	**None**
Queue Discipline	Specifies the method that the rate class uses to queue and dequeue traffic. Allowed settings are **Stochastic Fair Queue** and **Priority FIFO**.	Same as parent class if a parent class is specified; otherwise, **Stochastic Fair Queue**

Table 12.1 *Settings for configuring a rate class*

Before configuring the settings of a rate class, it is helpful to have a description of those settings.

Specifying a name

The first setting you configure for a rate class is the rate class name. Rate class names are case-sensitive and may contain letters, numbers, and underscores (_) only. Reserved keywords are not allowed.

Each rate class that you define must have a unique name. This setting is required.

To specify a rate class name, locate the **Name** box on the New Rate Class screen and type a unique name for the rate class.

Specifying a base rate

The **Base Rate** setting specifies the base throughput rate allowed for traffic that the rate class handles. Packets are generally not allowed to exceed the specified rate. You can specify the base rate in bits per second (bps), kilobits per second (Kbps), megabits per second (Mbps), or gigabits per second (Gbps). The default unit is bits per second. This setting is required.

The minimum base rate that you can configure is 296 bps.

◆ Note

These numbers are powers of 10, not powers of 2.

Specifying a ceiling rate

The **Ceiling Rate** setting specifies the absolute limit at which traffic is allowed to flow when bursting or borrowing. You can specify the ceiling rate in bits per second (bps), kilobits per second (Kbps), megabits per second (Mbps), or gigabits per second (Gbps). The default unit is bits per second.

If you specify a ceiling rate, the rate must be equal to or greater than the base rate. If you omit the ceiling rate or set it equal to the base rate, traffic throughput can never exceed the base rate.

Specifying a burst size

You use the **Burst Size** setting when you want to allow the rate of traffic flow that a rate class controls to exceed the base rate. Exceeding the base rate is known as ***bursting***. When you configure a rate class to allow bursting (by specifying a value other than **0**), the LTM system saves any unused bandwidth and uses that bandwidth later to enable the rate of traffic flow to temporarily exceed the base rate. Specifying a burst size is useful for smoothing out traffic patterns that tend to fluctuate or exceed the base rate, such as HTTP traffic.

The value of the **Burst Size** setting defines the maximum number of bytes that you want to allow for bursting. Thus, if you set the burst size to 5,000 bytes, and the rate of traffic flow exceeds the base rate by 1,000 bytes per second, then the LTM system allows the traffic to burst for a maximum of five seconds.

When you specify a burst size, the LTM system creates a burst reservoir of that size. A ***burst reservoir*** stores bandwidth that the LTM uses for bursting later. The burst reservoir becomes depleted as the rate of traffic flow exceeds the base rate, and is replenished as the rate of traffic falls below the base rate. The **Burst Size** value that you configure in a rate class thus represents:

- The maximum number of bytes that the rate class is allowed to transmit when the traffic-flow rate exceeds the base rate
- The maximum number of bytes that the LTM can replenish into the burst reservoir
- The amount of bandwidth initially available for bursting beyond the base rate

The burst size is measured in bytes. For example, a value of either **10000** or **10K** equals 10,000 bytes. The default value is **0**.

Depleting the burst reservoir

When the rate of traffic flow exceeds the base rate, the LTM system automatically depletes the burst reservoir, at a rate determined by the number of bytes per second that the traffic flow exceeds the base rate.

Continuing with our previous example in which traffic flow exceeds the base rate by 1,000 bytes per second, if the traffic-flow rate only exceeds the base rate for two seconds, then 2,000 bytes are depleted from the burst size and the maximum bytes available for bursting decreases to 3,000.

Replenishing the burst reservoir

When the rate of traffic flow falls below the base rate, the LTM system stores the unused bandwidth (that is, the difference between the base rate and the actual traffic-flow rate) in the burst reservoir. Later, the LTM system uses this bandwidth when traffic flow exceeds the base rate. Thus, the LTM system replenishes the burst reservoir whenever it becomes depleted due to traffic flow exceeding the base rate.

The size of the burst reservoir cannot exceed the specified burst size. For this reason, the LTM system replenishes the reservoir with unused bandwidth only until the reservoir reaches the amount specified by the

Burst Size setting. Thus, if the burst size is set to **5,000**, then the LTM system can store only 5,000 bytes of unused bandwidth for later use when the rate of traffic flow exceeds the base rate.

Note

Specifying a burst size does not allow the rate class to exceed its ceiling rate.

Specifying a non-zero burst size

The following example illustrates the behavior of the LTM system when you set the **Burst Size** setting to a value other than **0**.

This example shows throughput rates in units of bytes-per-second instead of the default bits-per-second. This is only to simplify the example. You can derive bytes-per-second from bits-per-second by dividing the bits-per-second amount by 8.

Suppose you configure the rate class settings with these values:

- Base rate: 1,000 bytes per second
- Ceiling rate: 4,000 bytes per second
- Burst size: 5,000 bytes

Consider the following scenario:

- **If traffic is currently flowing at 800 bytes per second**
  No bursting is necessary because the rate of traffic flow is below the base rate defined in the rate class.

  Because the traffic is flowing at 200 bytes per second lower than the base rate, the LTM system can potentially add 200 bytes of unused bandwidth to the burst reservoir. However, because no bursting has occured yet, the reservoir is already full at the specified 5,000 bytes, thus preventing the LTM system from storing the 200 bytes of unused bandwidth in the reservoir. In this case, the LTM system simply discards the unused bandwidth.
- **If traffic climbs to 1,000 bytes per second (equal to the base rate)**
  Still no bursting occurs, and there is no unused bandwidth.
- **If traffic jumps to 2,500 bytes per second**
  For each second that the traffic continues to flow at 2,500 bytes per second, the LTM system empties 1,500 bytes from the burst reservoir (the difference between the traffic flow rate and the base rate). This allows just over three seconds of bursting at this rate before the burst reservoir of 5,000 bytes is depleted. Once the reservoir is depleted, the LTM system reduces the traffic flow rate to the base rate of 1,000 bytes per second, with no bursting allowed.
- **If traffic drops back down to 800 bytes per second**
  No bursting is necessary, but now the LTM system can add the 200 bytes per second of unused bandwidth back into the burst reservoir because the reservoir is empty. If traffic continues to flow at 800 bytes per second,

the burst reservoir becomes fully replenished from 0 to 5,000 bytes in 25 seconds (at a rate of 200 bytes per second). If traffic stops flowing altogether, creating 1,000 bytes per second of unused bandwidth, then the LTM system adds 1,000 bytes per second into the burst reservoir, thus replenishing the reservoir from 0 to 5,000 bytes in only 5 seconds.

Borrowing bandwidth

In some cases, a rate class can borrow bandwidth from the burst reservoir of its parent class. For more information, see *Specifying a parent class*, following.

Specifying direction

Using the **Direction** setting, you can apply a rate class to client or server traffic. Thus, you can apply a rate class to traffic going to a client, to a server, or to both client and server. Possible values are **Any**, **Client**, and **Server**. The default value is **Any**.

Specifying direction is useful in cases where the nature of the traffic is directionally-biased. For example, if you offer an FTP service to external clients, you might be more interested in limiting throughput for those clients uploading files to your site than you are for clients downloading files from your site. In this case, you would select **Server** as the direction for your FTP rate class, because the **Server** value only applies your throughput restriction to traffic going from the client to the server.

Specifying a parent class

When you create a rate class, you can use the **Parent Class** setting to specify that the rate class has a parent class. This allows the rate class to borrow unused bandwidth from that parent class. A child class can borrow unused bandwidth from its parent, but a parent class cannot borrow from a child class. Borrowing is also not possible between two child classes of the same parent class or between two unrelated rate classes.

You specify a parent class by displaying the New Rate Class screen and selecting **Advanced**, and then selecting a rate class name in the **Parent Class** setting.

A parent class can itself have a parent, provided that you do not create a circular dependency. A ***circular dependency*** is a relationship where a rate class is a child of itself, directly or indirectly.

If a rate class has a parent class, the child class can take unused bandwidth from the parent class. The process occurs in this way:

- If the rate of traffic flow to which the child class is applied exceeds its base rate, the child class begins to deplete its burst reservoir as described previously.

- If the reservoir is empty (or no burst size is defined for the rate class), then the LTM system takes unused base-rate bandwidth from the parent class and gives it to the child class.
- If the unused bandwidth from the parent class is depleted, then the child class begins to use the reservoir of the parent class.
- If the reservoir of the parent class is empty (or no burst size is defined for the parent class), then the child class attempts to borrow bandwidth from the parent of the parent class, if the parent class has a parent class.
- This process continues until there is no remaining bandwidth to borrow or there is no parent from which to borrow.

Borrowing only allows the child to extend its burst duration; the child class cannot exceed the ceiling rate under any circumstance.

Note

Although the above description uses the term "borrowing," bandwidth that a child class borrows is not paid back to the parent class later, nor is unused bandwidth of a child class returned to its parent class.

Specifying a queue discipline

The **Queue Discipline** setting determines the method and order in which the LTM system dequeues packets.

A rate class supports two queue disciplines:

- **Stochastic Fair Queue**
  Stochastic Fair Queueing (SFQ) is a queueing method that queues traffic under a set of many lists, choosing the specific list based on a periodically-changing hash of the connection information. This results in traffic from the same connection always being queued in the same list. SFQ then dequeues traffic from the set of the lists in a round-robin fashion. The overall effect is that fairness of dequeueing is achieved because one high-speed connection cannot monopolize the queue at the expense of slower connections.
- **Priority FIFO**
  The ***Priority FIFO (PFIFO)*** queueing method queues all traffic under a set of five lists based on the Type of Service (ToS) field of the traffic. Four of the lists correspond to the four possible ToS values (**Minimum delay**, **Maximum throughput**, **Maximum reliability**, and **Minimum cost**). The fifth list represents traffic with no **ToS** value. The PFIFO method then processes these five lists in a way that attempts to preserve the meaning of the **ToS** field as much as possible. For example, a packet with the **ToS** field set to **Minimum cost** might yield dequeuing to a packet with the **ToS** field set to **Minimum delay**.

Managing rate classes

Once you have created a rate class, you can use the Configuration utility to list existing rate classes, view or modify the settings of a rate class, or delete a rate class.

To list existing rate classes

1. On the Main tab, expand **Local Traffic**.
2. Click **Rate Shaping**.
   This displays a list of existing rate classes and their setting values.
3. View the list of rate classes.

To view or modify a rate class

1. On the Main tab, expand **Local Traffic**.
2. Click **Rate Shaping**.
   This displays a list of existing rate classes.
3. Click a rate class name in the list.
   This displays the settings for that rate class.
4. Retain or modify any setting values. For information rate class settings, see *Configuring rate class settings*, on page 12-3.
5. Click **Update**.

To delete a rate class

1. On the Main tab, expand **Local Traffic**.
2. Click **Rate Shaping**.
   This displays a list of existing rate classes.
3. Locate a rate class name in the list, and to the left of the name, check the Select box.
4. At the bottom of the screen, click **Delete**.
   This displays a screen to confirm the deletion.
5. Click **Delete**.
   This removes the rate class.

13

Writing iRules

- Introducing iRules
- Creating iRules
- Controlling iRule evaluation
- Using statement commands
- Querying header or content data
- Manipulating header or content data
- Using utility commands
- Working with profiles
- Enabling session persistence with iRules
- Creating, managing, and using data groups

Introducing iRules

An ***iRule*** is a powerful and flexible feature within the BIG-IP® local traffic management (LTM) system that you can use to manage your network traffic. The iRules™ feature not only allows you to select pools based on header data, but also allows you to direct traffic by searching on any type of content data that you define. Thus, the iRules feature significantly enhances your ability to customize your content switching to suit your exact needs.

The remainder of this introduction presents an overview of iRules, lists the basic elements that make up an iRule, and shows some examples of how to use iRules to direct traffic to a specific destination such as a pool or a particular node.

What is an iRule?

An iRule is a script that you write if you want individual connections to target a pool other than the default pool defined for a virtual server. iRules allow you to more directly specify the pools to which you want traffic to be directed. Using iRules, you can send traffic not only to pools, but also to individual pool members, ports, or URIs.

The iRules you create can be simple or sophisticated, depending on your content-switching needs. Figure 13.1 shows an example of a simple iRule.

```
when CLIENT_ACCEPTED {
  if { [IP::addr [IP::client_addr] equals 10.10.10.10] } {
     pool my_pool
 }
}
```

Figure 13.1 *Example of an iRule*

This iRule is triggered when a client-side connection has been accepted, causing the LTM system to send the packet to the pool **my_pool**, if the client's address matches **10.10.10.10**.

Using a feature called the ***Universal Inspection Engine (UIE)***, you can write an iRule that searches either a header of a packet, or actual packet content, and then directs the packet based on the result of that search. iRules can also direct packets based on the result of a client authentication attempt.

iRules can direct traffic not only to specific pools, but also to individual pool members, including port numbers and URI paths, either to implement persistence or to meet specific load balancing requirements.

The syntax that you use to write iRules is based on the Tool Command Language (Tcl) programming standard. Thus, you can use many of the standard Tcl commands, plus a robust set of extensions that the LTM system provides to help you further increase load balancing efficiency.

For information about standard Tcl syntax, see **http://tmml.sourceforge.net/doc/tcl/index.html**. For a list of Tcl commands that have been disabled within the LTM system and therefore cannot be used when writing iRules, see Appendix B, *Disabled Tcl Commands*.

Basic iRule elements

iRules are made up of these basic elements:

- Event declarations
- Operators
- iRule commands

Event declarations

iRules are event-driven, which means that the LTM system triggers an iRule based on an event that you specify in the iRule. An ***event declaration*** is the specification of an event within an iRule that causes the LTM system to trigger that iRule whenever that event occurs. Examples of event declarations that can trigger an iRule are **HTTP_REQUEST**, which triggers an iRule whenever the system receives an HTTP request, and **CLIENT_ACCCEPTED**, which triggers an iRule when a client has established a connection.

For more information on iRule events, see *Specifying events*, on page 13-8.

Operators

An iRule operator compares two operands in an expression. In addition to using the Tcl standard operators, you can use the operators listed in Table 13.1.

Operator	Syntax

Relational operators	`contains`
	`matches`
	`equals`
	`starts_with`
	`ends_with`
	`matches_regex`

Logical operators	`not`
	`and`
	`or`

Table 13.1 *iRule operators*

For example, you can use the **contains** operator to compare a variable operand to a constant. You do this by creating an **if** statement that represents the following: "If the HTTP URI contains **aol**, send to pool **aol_pool**." Figure 13.2, on page 13-3 shows an iRule that performs this action.

```
when HTTP_REQUEST {
  if { [HTTP::uri] contains "aol" } {
     pool aol_pool
  } else {
     pool all_pool
 }
}
```

***Figure 13.2** An iRule based on the **contains** operator*

iRule commands

An iRule ***command*** within an iRule causes the LTM system to take some action, such as querying for data, manipulating data, or specifying a traffic destination. The types of commands that you can include within iRules are:

- **Statement commands**
  These commands cause actions such as selecting a traffic destination or assigning a SNAT translation address. An example of a statement command is **pool <name>**, which directs traffic to the named load balancing pool. For more information, see *Using statement commands*, on page 13-13.
- **Query commands**
  These commands search for header and content data. An example of a query command is **IP::remote_addr**, which searches for and returns the remote IP address of a connection. For more information on query commands, see *Querying header or content data*, on page 13-15.
- **Data manipulation commands**
  These commands perform data manipulation such as inserting headers into HTTP requests. An example of a data manipulation command is **HTTP::header remove <name>**, which removes the last occurrence of the named header from a request or response. For more information on data manipulation commands, see *Manipulating header or content data*, on page 13-24.
- **Utility commands**
  These commands are functions that are useful for parsing and manipulating content. An example of a utility command is **decode_uri <string>**, which decodes the named string using **HTTP URI** encoding and returns the result. For more information on using utility commands, see *Using utility commands*, on page 13-33.

Specifying traffic destinations and address translations

As described in the previous section, iRule commands instruct the LTM system to take direct action in some way. The following sections show examples of iRule commands that either direct traffic to a specific destination or assign translation addresses for SNAT implementation.

For detailed information on iRule commands, see these sections:

- *Using statement commands*, on page 13-13
- *Querying header or content data*, on page 13-15
- *Manipulating header or content data*, on page 13-24
- *Using utility commands*, on page 13-33

Selecting a load balancing pool

Once you have specified a query within your iRule, you can use the **pool** command to select a load balancing pool to which you want the LTM system to send a request. Figure 13.3 shows an example of this command.

```
when HTTP_REQUEST {
  set uri [HTTP::uri]
  if { $uri ends_with ".gif" } {
     pool my_pool
  } elseif { $uri ends_with ".jpg" } {
     pool your_pool
 }
}
```

Figure 13.3 *Example of the **pool** command within an iRule*

Selecting a specific server

As an alternative to the **pool** command, you can also write an iRule that directs traffic to a specific server. To do this, you use the **node** command. Figure 13.4 shows an example of this command.

```
when HTTP_REQUEST {
  if { [HTTP::uri] ends_with ".gif" } {
     node 10.1.2.200 80
 }
}
```

Figure 13.4 *Example of the **node** command within an iRule*

Selecting a pool of cache servers

You can create an iRule that selects a server from a pool of cache servers. Figure 13.5 shows an iRule that selects a server from a pool of cache servers.

```
when HTTP_REQUEST
   # This line specifies the expressions that determine whether the BIG-IP system sends
requests to the cache pool:
   if { [HTTP::uri] ends_with "html" or [HTTP::uri] ends_with "gif" } {
      pool cache_pool
      set key [crc32 [concat [domain [HTTP::host] 2] [HTTP::uri]]]
      set cache_mbr [persist lookup hash $key node]
      if { $cache_mbr ne "" } {
        # This line verifies that the request is not coming from the cache:
        if { [IP::addr [IP::remote_addr] equals $cache_mbr] }
          # This line sends the request from the cache to the origin pool:
             pool origin_pool
             return
          }
       }
       # These lines ensure that the persistence record is added for this host/URI:
       persist hash $key
    } else {
       pool origin_pool
    }
}
```

Figure 13.5 *Load balancing to a server in a cache pool*

Note that the BIG-IP system redirects URIs to a new cache member at the time that the BIG-IP system receives a request for the URI, rather than when the pool member becomes unavailable.

Redirecting HTTP requests

In addition to configuring an iRule to select a specific pool, you can also configure an iRule to redirect an HTTP request to a specific location, using the **HTTP::redirect** iRule command. The location can be either a host name or a URI.

For example, Figure 13.6 shows an iRule that is configured to redirect an HTTP response.

```
when HTTP_RESPONSE {
  if { [HTTP::status] contains "404"} {
    HTTP::redirect "http://www.siterequest.com/"
 }
}
```

Figure 13.6 *An iRule based on HTTP redirection*

Figure 13.7 shows an example of an iRule that redirects an HTTP request.

```
when HTTP_REQUEST {
  if { [HTTP::uri] contains "secure"} {
    HTTP::redirect "https://[HTTP::host][HTTP::uri]"
 }
}
```

Figure 13.7 *Another iRule based on HTTP redirection*

Assigning translation addresses for SNAT connections

The iRules feature includes the two statement commands **snat** and **snatpool.** Using the **snat** command, you can assign a specified translation address to an original IP address from within the iRule, instead of using the SNAT screens within the Configuration utility.

Using the **snatpool** command also assigns a translation address to an original IP address, although unlike the **snat** command, the **snatpool** command causes the LTM system to select the translation address from a specified SNAT pool that you previously created.

For more information on implementing SNATs, see Chapter 11, *Configuring SNATs and NATs*.

Creating iRules

You create an iRule using the Configuration utility.

To create an iRule

1. On the Main tab, expand **Local Traffic**.
2. Click i**Rules**.
   The iRules screen opens.
3. In the upper right corner, click **Create**.
4. In the **Name** box, type a 1- to 31-character name.
5. In the **Definition** box, type the syntax for your iRule.
6. Click **Finished**.

For detailed syntax information on writing iRules, see the remainder of this chapter.

Important

Once you have created an iRule, you need to configure a virtual server to reference the iRule. For information on configuring a virtual server to reference an iRule, see Chapter 2, ***Configuring Virtual Servers****.*

Controlling iRule evaluation

In a basic system configuration where no iRule exists, the LTM system directs incoming traffic to the default pool assigned to the virtual server that receives that traffic. However, you might want the LTM system to direct certain kinds of connections to other destinations. The way to do this is to write an iRule that directs traffic to that other destination, contingent on a certain type of event occurring. Otherwise, traffic continues to go to the default pool assigned to the virtual server.

iRules are therefore evaluated whenever an event occurs that you have specified in the iRule. For example, if an iRule includes the event declaration CLIENT_ACCEPTED, then the iRule is triggered whenever the LTM system accepts a client connection. The LTM system then follows the directions in the remainder of the iRule to determine the destination of the packet.

Configuration prerequisites

Before the LTM system can evaluate the iRule that you have written, you must do the following:

- **Assign the iRule to a virtual server.**
  When an iRule is assigned to virtual server, this means that the virtual server *references* the iRule, similar to the way that a virtual server references a pool or a profile.
- **Ensure that the virtual server references the appropriate profile.**
  For example, if your iRule includes the event declaration **HTTP_REQUEST**, then the LTM system only evaluates the iRule if the virtual server references an **http** profile type.

Note

*When assigning an iRule that specifies the event **HTTP_REQUEST**, make sure that the virtual server references the appropriate profile type.*

For information on assigning iRules and profiles to virtual servers, see Chapter 2, *Configuring Virtual Servers*.

Specifying events

The iRules feature includes several types of event declarations that you can make in an iRule. Specifying an event declaration determines when the LTM system evaluates the iRule. The following sections list and describe these event types. Also described is the concept of iRule context and the use of the **when** keyword.

Event types

The iRule command syntax includes several types of event declarations that you can specify within an iRule:

- Global events
- HTTP events
- SSL events
- Authentication events

Table 13.2 lists and describes the events that you can declare in an iRule for each of these event types.

iRule Event	Description
Global Events	
CLIENT_ACCEPTED	Triggered when a client establishes a connection.
CLIENT_DATA	Triggered when a client receives new data while the connection is in **collect** state.
LB_SELECTED	Triggered when the LTM system has selected a target node.
LB_FAILED	Triggered when a connection to the server was unable to complete. This might occur if the pool has no available members or a selected pool member is otherwise not available.
SERVER_CONNECTED	Triggered when the system establishes a connection with the target node.
SERVER_DATA	Triggered when the system has received new data from the target node while the connection is in **hold** state.
RULE_INIT	Triggered when you add or modify an iRule. You use this event to initialize global variables that you are using within iRules.
CLIENT_CLOSED	Triggered when the client's connection is closed.
SERVER_CLOSED	Triggered when the server's connection is closed.
HTTP Events	
HTTP_CLASS_SELECTED	Triggered when an HTTP request matches an HTTP class. You can use the **HTTP::class** command to extract the matching class name.
HTTP_REQUEST	Triggered when the system fully parses a complete client request header (that is, the method, URI, version and all headers, not including the body). Note that this event is evaluated on every request, rather than on the first request only. For this reason, you might want to consider dividing the iRule into a CLIENT_ACCEPTED event and an HTTP_REQUEST event, when using both Layer 4 and Layer 7 iRule commands.
HTTP_REQUEST_DATA	Triggered whenever an **HTTP::collect** command finishes running, after collecting the requested amount of request data. For more information, see *Querying HTTP headers and content*, on page 13-20.
HTTP_REQUEST_SEND	Triggered just before a request is sent to a server. This is a server-side event.
HTTP_RESPONSE	Triggered when the system parses all of the response status and header lines from the server response.
HTTP_RESPONSE_DATA	Triggered whenever an **HTTP::collect** command finishes running on the server side of a connection, after collecting the requested amount of response data. Also triggered if the server closes the connection before the **HTTP:collect** command finishes running. For more information, see *Querying HTTP headers and content*, on page 13-20.

Table 13.2 *Event declarations for iRules*

iRule Event	Description
HTTP_RESPONSE_CONTINUE	Triggered whenever the system receives a **100 Continue** response from the server.
CACHE_REQUEST	Triggered when the system receives a request for a cached object. Used to override default behavior.
CACHE_RESPONSE	Triggered immediately prior to sending a cache response. Used to override default behavior.
SSL Events	
CLIENTSSL_HANDSHAKE	Triggered when a client-side SSL handshake is completed.
CLIENTSSL_CLIENTCERT	Triggered when the system adds an SSL client certificate to the client certificate chain. The LTM system can retrieve the X509 certificate and its X509 issuer with the **SSL::cert** and **SSL::cert issuer** commands.
SERVERSSL_HANDSHAKE	Triggered when a server-side SSL handshake is completed.
Authentication Events	
AUTH_FAILURE	Triggered when an unsuccessful authorization operation is completed. A default handler for this event is associated with each of the authentication profiles, and causes the system to close the connection.
AUTH_ERROR	Triggered when an error occurs during authorization. A default handler for this event is associated with each of the authentication profiles, and causes the system to close the connection. The associated authentication session ID is invalidated and the user should immediately discard the session ID upon receipt of this event.
AUTH_WANTCREDENTIAL	Triggered when an authorization operation needs an additional credential. See also the description of the **AUTH::wantcredential_prompt.** command in section *Querying authentication data*, on page 13-23. A default handler for this event is associated with each of the authentication profiles, and causes the system to close the connection unless it can obtain the desired credential. Typically this implies that the protocol layer that provides the credential has also not yet obtained the credential, because the system did not enable the necessary authentication protocol. Each of the authentication profiles contains appropriate default handlers for its respective protocol.
AUTH_SUCCESS	Triggered when a successful authorization has completed all of the required authentication services.

Table 13.2 *Event declarations for iRules*

iRule context

For every event that you specify within an iRule, you can also specify a context, denoted by the keywords **clientside** or **serverside**. Because each event has a default context associated with it, you need only declare a context if you want to change the context from the default.

For example, Figure 13.8 shows **my_iRule1**, which includes the event declaration **CLIENT_ACCEPTED**, as well as the iRule command **IP::remote_addr**. In this case, the IP address that the iRule command returns is that of the client, because the default context of the event declaration **CLIENT_ACCEPTED** is **clientside**.

```
when CLIENT_ACCEPTED {
  if { [IP::addr [IP::remote_addr] equals 10.1.1.80] } {
     pool my_pool1
 }
}
```

Figure 13.8 *An iRule that uses default* ***clientside*** *context*

Similarly, if you include the event declaration **SERVER_CONNECTED** in an iRule as well as the iRule command **IP::remote_addr**, the IP address that the iRule command returns is that of the server, because the default context of the event declaration **SERVER_CONNECTED** is **serverside**.

Figure 13.8 shows what happens when you write an iRule that uses the default context when processing iRule commands. You can, however, explicitly specify the **clientside** and **serverside** keywords to alter the behavior of iRule commands.

Continuing with the previous example, Figure 13.9 shows the event declaration **SERVER_CONNECTED** and explicitly specifies the **clientside** keyword for the iRule command **IP::remote_addr**. In this case, the IP address that the iRule command returns is that of the client, despite the serverside default context of the event declaration.

```
when SERVER_CONNECTED {
  if { [IP::addr [IP::addr [clientside {IP::remote_addr}] equals 10.1.1.80] } {
     discard
  }
 }
```

Figure 13.9 *An iRule that explicitly specifies context*

For more information on specifying iRule context, see *Using statement commands* on this page.

Using the when keyword

You make an event declaration in an iRule by using the **when** keyword, followed by the event name. The previous figure shows an example of an event declaration in an iRule.

Listing iRules on a virtual server

When you assign multiple iRules as resources for a virtual server, it is important to consider the order in which you list them on the virtual server. This is because the LTM system processes duplicate iRule events in the order that the applicable iRules are listed. An iRule event can therefore terminate the triggering of events, thus preventing the LTM system from triggering subsequent events.

Note

If an iRule references a profile, the LTM system processes this type of iRule last, regardless of its order in the list of iRules assigned to a virtual server.

Using statement commands

Some of the commands available for use within iRules are known as statement commands. ***Statement commands*** enable the LTM system to perform a variety of different actions. For example, some of these commands specify the pools or servers to which you want the LTM system to direct traffic. Other commands specify translation addresses for implementing SNAT connections. Still others specify objects such as data groups or a persistence profiles.

Table 13.3 lists and describes statement commands that you can use within iRules.

Statement Command	Description
clientside {<iRule commands>}	Causes the specified iRule commands to be evaluated under the client-side context. This command has no effect if the iRule is already being evaluated under the client-side context.
discard	Causes the current packet or connection (depending on the context of the event) to be discarded. This statement must be conditionally associated with an **if** statement.
drop	Same as the **discard** command.
event [<name>] disable | disable all	Discontinues evaluating the specified iRule event, or all iRule events, on this connection. However, the iRule continues to run.
forward	Sets the connection to forward IP packets.
if { <expression> } { <statement_command> } elseif { <expression> } { <statement_command> }	Asks a true or false question and, depending on the answer, takes some action. Note that the maximum number of **if** statements that you can nest in an iRule is 100.
log [<facility>.<level>] <message>	Generates and logs the specified message to the Syslog facility. The statement does this by performing variable expansion on the message as defined for the **Header Insert** HTTP profile attribute. If not used appropriately, a **log** statement can produce large amounts of output.
matchclass <data-group operator name>	Specifies a data group. Often used within an **if** statement and in conjunction with the **contains** operator.
[use] node <addr> [<port>]	Causes the identified server node to be used directly, thus bypassing any load-balancing.
peer { <iRule commands> }	Causes the specified iRule commands to be evaluated under the peer's (opposite) context.

Table 13.3 *iRule statement commands*

Statement Command	Description
persist <persistence_type>	Causes the LTM system to use the named persistence profile to persist the connection. For more information on using iRules to enable persistence, see *Working with profiles*, on page 13-37.
[use] pool <pool_name> [member <addr> [<port>]]	Causes the LTM system to load balance traffic to the named pool. This statement must be conditionally associated with an **if** statement. Optionally, you can specify a specific pool member to which you want to direct the traffic.
[use] rateclass <rate_class>	Causes the LTM system to select the specified rate class to use when transmitting packets.
reject	Causes the connection to be rejected, returning a reset as appropriate for the protocol.
return	Terminates execution of the iRule event .
serverside { <iRule commands> }	Causes the specified iRule commands to be evaluated under the server-side context. This command has no effect if the iRule is already being evaluated under the server-side context.
[use] snat <addr> [<port>] | none	Causes the LTM system to assign the specified translation address to one or more origin IP addresses specified in the iRule.
[use] snatpool <snatpool_name> | none	Causes the pool of addresses identified by **<snatpool_name>** to be used as translation addresses to create a SNAT.

Table 13.3 *iRule statement commands*

Querying header or content data

The iRules feature includes several commands that are specifically designed to allow you to run queries on the following:

- Node status
- Link Layer headers
- IP headers
- TCP headers and content
- UDP headers and content
- SSL headers in HTTP requests
- Authentication data
- Statistics data

Querying for node status

You can query for the status of a node by using the **LB::status** command. Table 13.4 describes this command.

iRule command	Description
LB::status LB::status node <address> LB::status pool <pool name> member <IP address> <port>	Returns the status of a node. Possible values are **up**, **down**, **session_enabled**, and **session_disabled**. If you supply no arguments, returns the status of the currently-selected node.

Table 13.4 *iRule commands for querying node status*

Querying Link Layer headers

You can select a pool by specifying Link Layer header information. Table 13.5 lists and describes these commands.

iRule Command	Description
LINK::vlan_id	Returns the VLAN tag of the packet.
LINK::vlan_qos	Returns the VLAN Quality of Service (QoS) value of the packet.
LINK::lasthop	Returns the last hop MAC address.
LINK::nexthop	Returns the next hop MAC accress.

Table 13.5 *iRule commands for querying Link Layer headers headers*

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 LTM system can apply an iRule that sends the traffic to different pools of servers based on the QoS level within a packet.

To configure an iRule to select a pool based on the QoS level of a packet, you can use the **LINK::vlan_qos** iRule command, as you can see in the example in Figure 13.10.

```
when CLIENT_ACCEPTED {
  if { [LINK::qos] > 2 } {
     pool fast_pool
  } else {
     pool slow_pool
 }
}
```

Figure 13.10 *An iRule based on a Quality of Service (QoS) level*

For more information on setting QoS values on packets, see *Manipulating Link Layer data*, on page 13-24 and Chapter 4, *Configuring Load Balancing Pools*.

Querying IP packet headers

You can select a pool by querying for IP packet header information. Table 13.6 lists and describes these commands.

iRule Command	Description
IP::remote_addr	Returns the remote IP address of a connection.
IP::local_addr	Returns the local IP address of a connection.
IP::client_addr	Returns the client IP address of a connection. This command is equivalent to the command **clientside { IP::remote_addr }**.
IP::server_addr	Returns the server's IP address. This command is equivalent to the command **serverside { IP::remote_addr }**. Will return **0** if the load-balancing decision has not occurred.
IP::protocol	Returns the IP protocol value.
IP::tos	Returns the value of the IP protocol's Type of Service (ToS) field.
IP::ttl	Returns the TTL for an inbound IPv4 or IPv6 packet from the peer.
IP::idle_timeout	Returns or sets the idle timeout value.
IP::ttl	Returns the time-to-live (TTL) value for an inbound IPv4 or IPv6 packet.

Table 13.6 *iRule commands for querying IP packet headers*

As you can see by the preceding table, the specific types of IP packet header data that you can query for within an iRule are:

- IP addresses

- Protocol numbers
- ToS levels
- Idle timeout values
- TTL values

Specifying an IP address

You can specify the **IP::remote_addr** or **IP::local_addr** command within an iRule to select a pool. For example, you can load balance traffic based on part of the client's IP address. You can also specify the **IP::client_addr** and **IP::server_addr** commands.

Figure 13.11 shows an iRule that implements the preceding statements. In this example, all client requests with the first byte of their source address equal to **206** are directed to a pool named **clients_from_206** pool. All other requests are directed to a pool named **other_clients_pool.**

```
when CLIENT_ACCEPTED {
  if { [IP::addr [IP::remote_addr] equals 206.0.0.0/255.0.0.0] } {
     pool clients_from_206
  } else {
     pool other_clients_pool
 }
}
```

***Figure 13.11** An iRule based on the **IP::remote_addr** command*

Specifying an IP protocol number

The LTM system includes an iRule command, **IP::protocol**, that you can use to select a pool based on an IP protocol number.

To configure an iRule to select a pool based on an IP protocol number, use the syntax shown in the example in Figure 13.12.

```
when CLIENT_ACCEPTED {
  if { [IP::protocol] == 6 } {
     pool tcp_pool
  } else {
     pool slow_pool
 }
}
```

***Figure 13.12** An iRule based on an IP protocol number*

Specifying a 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 LTM system can apply an iRule that sends the traffic to different pools of servers based on the ToS level within a packet.

The command that you use to set the ToS level on a packet is **IP::tos**.

To configure an iRule to select a pool based on the ToS level of a packet, you can use the **IP::tos** iRule command, as shown in the example in Figure 13.13.

```
when CLIENT_ACCEPTED {
  if { [IP::tos] == 16 } {
     pool telnet_pool
  } else {
     pool slow_pool
 }
}
```

Figure 13.13 *An iRule based on a Type of Service (ToS) level*

For more information on setting ToS values on packets, see *Manipulating IP headers*, on page 13-24 and Chapter 4, *Configuring Load Balancing Pools*.

Specifying an idle timeout value

Using the **IP::idle_timeout** command within an iRule, you can specify an idle timeout value as the criteria for selecting the pool to which you want the LTM system to send traffic.

Querying UDP headers and content

You can select a pool by specifying UDP header or content information. Table 13.7 lists and describes these commands.

iRule Command	Description
UDP::remote_port	Returns the remote's UDP port/service number.
UDP::local_port	Returns the local UDP port/service number.
UDP::client_port	Returns the client's UDP port/service number. Equivalent to the command **clientside { UDP::remote_port }**.
UDP::server_port	Returns the server UDP port/service number. Equivalent to the command **serverside { UDP::remote_port }**.
UDP::payload [<size>]	Returns the current UDP payload content.
UDP::payload length	Returns the amount of UDP payload content in bytes.

Table 13.7 *iRule commands for querying UDP headers or content*

Querying TCP headers and content

You can select a pool by specifying TCP header or content information. Table 13.8 lists and describes these commands.

iRule Command	Description
TCP::remote_port	Returns the remote TCP port/service number.
TCP::local_port	Returns the local TCP port/service number.
TCP::client_port	Returns the client's TCP port/service number. Equivalent to the command **clientside { TCP::remote_port }**.
TCP::server_port	Returns the server TCP port/service number. Equivalent to the command **serverside { TCP::remote_port }**.
TCP::payload [<size>]	Returns the accumulated TCP data content.
TCP::payload_length	Returns the amount of accumulated TCP data content in bytes.
TCP::rtt	Returns the smoothed round-trip time estimate for a TCP connection.
TCP::mss	Returns the on-wire Maximum Segment Size (MSS) for a TCP connection.
TCP::unused_port	Returns an unused TCP port for the specified IP tuple, using the value of <hint_port> as a starting point.
TCP::offset	Returns the position in the TCP data stream in which the collected TCP data starts.

Table 13.8 *iRule commands for querying TCP headers or content*

For example, you might want an iRule 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**".

To accomplish this, you can write an iRule using the **TCP::payload** command. Figure 13.14 shows an iRule that illustrates this example.

```
when CLIENT_ACCEPTED {
  TCP::collect 15
}
when CLIENT_DATA {
  if { [TCP::payload 15] contains "XYZ" } {
     pool xyz_servers
  } else {
     pool web_servers
 }
}
```

Figure 13.14 *Sample iRule using a TCP request string command*

Querying HTTP headers and content

You can select a destination by specifying HTTP header or content information. Table 13.9 lists and describes these commands. Note that this list does not include commands for querying SSL-related headers in HTTP requests. For information on querying SSL headers, see *Querying SSL headers of HTTP requests*, on page 13-22.

iRule Command	Description
HTTP::request	Returns the raw request header string. You can access the request payload using the **HTTP::collect** command.
HTTP::retry <request>	Resends a request to a server. This command applies to server-side iRules.Must be a well-formed and complete request header. Behaves like other requests and generates all client-side events such as **HTTP_REQUEST**
HTTP::class	Returns the HTTP class selected by the HTTP selector.
HTTP::header [value] <name>	Returns value of the HTTP header named **<name>**. You can omit the **<value>** argument if the header name does not collide with any of the subcommands.
HTTP::header names	Returns a list of all the headers present on the request or response.
HTTP::header count	Returns the number of HTTP headers present on the request or response.
HTTP::header at <index>	Returns the HTTP header that the system finds at the zero-based index value.
HTTP::header exists <name>	Returns true if the named header is present on the request or response.
HTTP::method	Returns the type of HTTP request method.
HTTP::status	Returns the response status code.
HTTP::version ["0.9" | "1.0" | "1.1"]	Returns or sets the HTTP version of the request or response.
HTTP::username	Returns the user name part of the HTTP basic authorization.
HTTP::password	Returns the password part of the HTTP basic authorization.
HTTP::path [<string>]	Returns the path part of the HTTP request.
HTTP::uri [<string>]	Returns the complete URI of the request.
HTTP::query [<string>]	Returns the query part of the HTTP request.
HTTP::is_redirect	Returns a **true** value if the response is a certain type of redirect.
HTTP::is_keepalive	Returns a **true** value if this is a **Keep-Alive** connection.
HTTP::collect [<length>]	Collects the amount of data that you specify with the **[length]** argument. When the system collects the specified amount of data, it calls the Tcl event **HTTP_REQUEST_DATA** or **HTTP_RESPONSE_DATA**. Use great caution when omitting the value of the content length. Even though this is allowed in certain cases; doing so or using a value larger than the size of the actual length can stall the connection.
HTTP::release	Releases the collected data. Unless a subsequent **HTTP ::collect** command was issued, there is no need to use the **HTTP::release** command inside of the **HTTP_REQUEST_DATA** and **HTTP_RESPONSE_DATA** events, since in these cases, the data is implicitly released.
HTTP::payload [<size>]	Returns the content that the **HTTP::collect** command has collected thus far. If you do not specify a size, the system returns the collected content.
HTTP::payload length	Returns the size of the content that the command has collected thus far, not including the HTTP headers.
HTTP::payload replace <offset> <length> <string>	Replaces the amount of content that you specified with the **<length>** argument, starting at **<offset>** with **<string>**.
CACHE::hits	Returns the document cache hits.

Table 13.9 *iRule commands for querying HTTP headers and content*

iRule Command	Description
CACHE::age	Returns the age of the document in the cache, in seconds.
CACHE::headers	Returns the HTTP headers of the cache response. This command can only be used with the **CACHE_RESPONSE** event.
CACHE::payload	Returns the HTTP payload of the cache response. This command can only be used with the **CACHE_RESPONSE** event.
URI::protocol <string>	Extracts the protocol part from the URI string that you specify.
URI::basename <string>	Extracts the basename part from the URI string that you specify.
URI::path <string>	Extracts the path from the URI string that you specify.
URI::query <string>	Extracts the query part from the URI string that you specify.
URI::host <string>	Extracts the host part from the URI string that you specify.
URI::compare <uri1> <uri2>	Compares URIs as recommended by RFC 2616 section 3.2.3.
URI::decode <string>	Returns the decoded URI string.
URI::encode <string>	Returns the encoded URI string
URI::port <string>	Extracts the port part from the URI string that you specify.

Table 13.9 *iRule commands for querying HTTP headers and content*

For example, you may want an iRule 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, if the packet data contains an HTTP request with a URI starting with **/abc**, then load balance using the pool **abc_servers**. Otherwise, load balance using the virtual server's default pool."

Figure 13.15 shows an iRule that illustrates this example.

```
when HTTP_REQUEST {
  if { [HTTP::uri] ends_with "cgi" } {
     pool cgi_pool
  } elseif { [HTTP::uri] starts_with "/abc" } {
     pool abc_servers
 }
}
```

Figure 13.15 *Sample iRule using an HTTP request string command*

Querying SSL headers of HTTP requests

You can select a destination based on data that resides in the SSL headers of an HTTP request. Table 13.10 lists and describes these commands.

iRule Command	Description
SSL::mode	In a client-side context, returns one of **require**, **request**, **ignore**, or **auto**. In a server-side context, returns one of **require** or **ignore**.
SSL::cert <index>	Returns the index of the X509 SSL certificate in the peer certificate chain, where index is a value greater than or equal to zero. A value of zero denotes the first certificate in the chain, a value of one is the next, and so on. This command is currently applicable only under a client-side context and returns an error within a server-side context.
SSL::cert issuer <index>	Returns the issuer certificate of the index of the X509 SSL certificate in the peer certificate chain, where index is a value greater than or equal to zero. A value of zero denotes the first certificate in the chain, a value of one is the next, and so on. This command is currently applicable only under a client-side context and returns an error within a server-side context.
SSL::cert count	Returns the total number of certificates that the peer has offered.
SSL::verify_result	Returns the result code from peer certificate verification using the same values as the OpenSSL **SSL_get_verify_result()** function.
SSL::cipher name	Returns the current SSL cipher version using the format of the OpenSSL **SSL_CIPHER_get_version()** function.
SSL::cipher version	Returns the current SSL cipher version using the format of the OpenSSL **SSL_CIPHER_get_version()** function.
SSL::cipher bits	Returns the number of secret bits that the current SSL cipher used, using the format of the OpenSSL **SSL_CIPHER_get_bits()** function
SSL::current_sessionid	Returns the SSL session ID currently negotiated, or a value of **-1**, if no session ID exists.
SSL::modssl_sessionid_headers [<option>+]	Returns a list of fields that the system is to add to the HTTP headers in order to emulate **modssl** behavior. The return type is a Tcl list that the system then interprets as a header name/header value pair. The options that you can specify with this command are **initial** and **current**. For information on ModSSL emulation, see Chapter 7, *Managing SSL Traffic*.

Table 13.10 *iRule commands for querying SSL headers in HTTP requests*

Querying authentication data

You can select a destination based on authentication data. Table 13.11 lists and describes these commands.

iRule Command	Description
AUTH::wantcredential_prompt <authid>	Returns the authorization session authid's credential prompt string that the system last requested (when the system generated an AUTH_WANTCREDENTIAL event), for example, **Username:**. This command is especially helpful in providing authentication services to interactive protocols (for example, **telnet** and **ftp**), where the actual text prompts and responses may be directly communicated with the remote user.
AUTH::wantcredential_prompt_style <authid>	Returns the authorization session authid's credential prompt style that the system last requested (when the system generated an AUTH_WANTCREDENTIAL event). This value is either **echo_on, echo_off**, or **unknown**. This command is especially helpful in providing authentication services to interactive protocols (or example, **telnet** and **ftp**), where the actual text prompts and responses may be directly communicated with the remote user.
AUTH::wantcredential_type <authid>	Returns the authorization session authid's credential type that the system last requested (when the system generated an AUTH_WANTCREDENTIAL event). This value is either **username**, **password**, **x509**, **x509_issuer**, or **unknown**, based upon the system's assessment of the credential prompt string and style.
AUTH::status <authid>	Returns a value of **success**, **failure**, **error**, or **not-authed**, based on the result of the most recent authorization that the system performed for the authorization session authid.
AUTH::ssl_cc_ldap_username <authid>	Returns the user name that the system retrieved from the LDAP database from the last successful client certificate-based LDAP query for the authorization session authid. The system returns an empty string if the last successful query did not perform a successful client certificate-based LDAP query, or if no query has yet been performed.
AUTH::ssl_cc_ldap_status <authid>	Returns the status from the last successful client certificate-based LDAP query for the authorization session authid. The system returns an empty string if the last successful query did not perform a client certificate-based LDAP query, or if no query has yet been performed.

Table 13.11 *iRule commands for querying authentication data*

Querying for statistics data

The iRule command **STATS::get** retrieves a value of a field that is named in a Statistics profile. Table 13.12 describes this command.

iRule Command	Description
STATS::get <profile> <field>	Retrives the value of the named field in the named Statistics profile.

Table 13.12 *iRule command for querying Statistics data*

Manipulating header or content data

The iRules feature includes several commands that are specifically designed to manipulate certain types of data. Data manipulation in this case refers to inserting, replacing, and removing data, as well as setting certain values found in headers and cookies.

The types of headers and content that you can manipulate with iRules are:

- Link Layer data
- IP headers
- TCP headers
- HTTP headers and cookies
- SSL headers
- Statistical data

Manipulating Link Layer data

Using the command described in Table 13.13, you can set the QoS level for transmitting a packet.

iRule Command	Description
LINK::vlan_qos	Set the VLAN QoS level that you want the system to use when transmitting the packet.

Table 13.13 *iRule command for manipulating Link Layer data*

Manipulating IP headers

Using the command described in Table 13.14, you can set the ToS level for transmitting the packet.

iRule Command	Description
IP::hops	Finds the nearest, next-highest power of two in the range (such as 64, 128, 255) and subtracts the value retrieved by the **IP:ttl** command. With the **IP::hops** command, you can passively estimate the number of hops between a system and its peer. A hop of **0** indicates that the client is on the local network. For example, if the TTL value equals **55**, the number of estimated hops is 9 (64 minus 55). If the TTL value equals **127**, the number of estimated hops is 1 (128 minus 127).
IP::tos	Sets the IP ToS level that you want the system to use when transmitting the packet.

Table 13.14 *iRule command for manipulating IP header data*

Manipulating TCP headers and content

Using the commands described in Table 13.15, you can manipulate TCP headers and content data.

iRule Command	Description
TCP::collect <length>	Causes TCP to start collecting the specified amount of content data.
TCP::release	Causes TCP to resume processing the connection and to flush collected data.
TCP::payload replace <offset> <length> <data>	Replaces collected payload with the given data.
TCP::respond <data>	Sends the named data directly to the peer. This command is used to complete a protocol handshake with an iRule.
TCP::close	Closes the connection.

Table 13.15 *iRule command for manipulating TCP content data*

Manipulating HTTP headers, content, and cookies

There are several iRule commands that you can use in your iRules to manipulate HTTP header data, content data, and cookies.

Manipulating HTTP headers and content

You can use a number of iRule commands to manipulate HTTP headers and content. For example, you can use the **HTTP::header** commands to insert headers that specify X509 certificate field values into HTTP requests.

Table 13.16 lists the iRule commands that you can use to manipulate headers in HTTP requests and responses. For information on extracting X509 certificate field values that you can use with the **HTTP::header** commands, see *Extracting X509 certificate field values*, on page 13-29.

iRule Command	Description
HTTP::disable	Puts HTTP traffic filtering into **passthrough** mode. This command is useful when using an HTTP profile with an application that proxies data over HTTP. One use of this command is when you need to tunnel PPP over HTTP and disable HTTP processing once the connection has been established.
HTTP::header insert ["lws"] <name> <value>	Inserts the named HTTP header and its value into the end of the HTTP request or response. If you specify "lws", the system adds linear white space to long header values.
HTTP::header insert ["lws"] {n1, v1, n2, v2, n3, v3, …}	Passes a Tcl list to insert into a header. In such cases, the system treats the list as a list of name/value pairs. If you specify "lws", the system adds linear white space to long header values.
HTTP::header [value] <name> <string>	Sets the value of the named header. If the header is present, the command replaces the header; otherwise, the command adds the header. You can omit the **<value>** argument if the header name does not collide with any other values.

Table 13.16 *iRule commands for manipulating headers in HTTP requests and responses*

iRule Command	Description
HTTP::header replace <name> [<string>]	Replaces the last occurrence of the named header with the string **<string>**. This command performs a header insertion if the header was not present.
HTTP::header remove <name>	Removes the last occurrence of the named header from the request or response.
HTTP::close	Inserts a **Connection: Close** header and close the HTTP connection.
HTTP::redirect <url>	Redirects a HTTP request or response to the specified URL. Note that this command sends the response to the client immediately. Therefore, you cannot specify this command multiple times in an iRule, nor can you specify any other commands that modify header or content, after you specify this command.
HTTP::respond <status code> [content <content Value>] [<Header name> <Header Value>]+	This is a powerful API that allows users to generate or rewrite a client request or a server response. When the system runs the command on the client side, it sends the response to the client without any load balancing taking place. If the system runs the command on the server side, the content from the actual server is discarded and replaced with the information provided to this API. Note that because the system sends the response data immediately after this iRule runs, we recommend that you not run any more iRules after this API.
HTTP::header insert_modssl_fields {options}	Inserts a header into an HTTP request. This command name is a misnomer because it does not insert a ModSSLheader. Instead, the command inserts a header that specifies, for a client connection, either the source IP address, the port number, or both. If no options are specified, the command inserts the header **ClientIPAddress: <addr>:<service>**. If the optional argument **addr** or **service** is specified, the command inserts either the header **ClientIPAddress: <addr>** or the header **ClientTCPService: <service>**, respectively.
HTTP::header sanitize <header name>+	Removes all but the headers you specify. The exception to this is some essential HTTP headers.
HTTP::request_num	Returns the number of HTTP requests that a client made on the connection.
CACHE::disable	Disables the caching for this request. You can use this command with the **HTTP_RESPONSE** and **HTTP_REQUEST** events.
CACHE::enable	Forces the document to be cached. You can use his command with the **HTTP_RESPONSE** and **HTTP_REQUEST** events. You can also use this command to cache non-GET requests.
CACHE::expire	Forces the document to be revalidated from the server.
CACHE::userkey <key string>	Allows users to add user defined values to the key. For example you can use this rule to store different cached content for dialup and broad band by using the MTU as a user defined key string.
CACHE::useragent <string>	Overrides the **useragent <string>** value used by the cache to store the cached content. This can be used to group various user agent values into a single group to minimize duplicated cached content. This value must be specified in an **HTTP_REQUEST** event.
CACHE::uri <string>	Overrides the URI value used by the cache to store the cached content. This can be used to group various URIs with parameters that point to the same document. This value must be specified in an **HTTP_REQUEST** event.

Table 13.16 *iRule commands for manipulating headers in HTTP requests and responses*

iRule Command	Description
CACHE::accept_encoding <string>	Overrides the **accept_encoding** value used by the cache to store the cached content. This can be used to group various user encoding values into a single group to minimize duplicated cached content. This value must be specified in an **HTTP_REQUEST** event.
CACHE::priority <1 .. 10>	Adds a priority to cached documents. The priority can be between 1 and 10 inclusive. This allows users to designate documents that are costly to produce as being more important than others. An example of this is a compressed document.
ONECONNECT::detach [enable | disable]	When enabled, detaches the client-side and server-side connection.
ONECONNECT::reuse [enable | disable]	When enabled, marks whether a connection can be re-used in a connection pool.
COMPRESS::enable	This command enables compression for the current HTTP response. Note that when using this command, you must set the HTTP profile setting **Compression** to **Selective**. You can only use this command in the context of the iRule events HTTP_REQUEST, HTTP_REQUEST_DATA, and HTTP_RESPONSE.
COMPRESS::disable	This command enables compression for the current HTTP response. Note that when using this command, you must set the HTTP profile setting **Compression** to **Selective**.You can only use this command in the context of the iRule events HTTP_REQUEST, HTTP_REQUEST_DATA, and HTTP_RESPONSE.
COMPRESS::buffer_size <value>	Sets the compression buffer size, described in Chapter 6, *Managing HTTP and FTP Traffic*. You can only use this command in the context of the iRule events HTTP_REQUEST, HTTP_REQUEST_DATA, and HTTP_RESPONSE.
COMPRESS::gzip memory_level <level>	Sets the **gzip** memory level, described in Chapter 6, *Managing HTTP and FTP Traffic*. You can only use this command in the context of the iRule events HTTP_REQUEST, HTTP_REQUEST_DATA, and HTTP_RESPONSE
COMPRESS::gzip window_size <size>	Sets the **gzip** window size, described in Chapter 6, *Managing HTTP and FTP Traffic*. You can only use this command in the context of the iRule events HTTP_REQUEST, HTTP_REQUEST_DATA, and HTTP_RESPONSE.
COMPRESS::gzip level <level>	Specifies the amount and rate of compression.
COMPRESS::method prefer ['gzip' | 'deflate']	Specifies the preferred compression algorithm.

Table 13.16 *iRule commands for manipulating headers in HTTP requests and responses*

As an option, the value of a header that you insert into an HTTP request can be the result of an SSL query command. To do this, specify an SSL query command as an argument to an **HTTP::header insert** command. For information on SSL query commands, see *Querying SSL headers of HTTP requests*, on page 13-22.

◆ Note

*Using **HTTP::header insert** or **HTTP::remove** commands in an iRule overrides the **Header Insert** and **Header Erase** settings in the corresponding HTTP profile.*

Manipulating HTTP cookies

Table 13.17 lists the iRule commands that you can use to manipulate cookies in HTTP requests and responses.

iRule Command	Description
Commands for request messages	
HTTP::cookie names	Returns the names of all the cookies present in the HTTP header.
HTTP::cookie count	Returns the number of cookies present in the HTTP header.
HTTP::cookie [value] <name> [string]	Sets or gets the cookie value of the given name. You can omit the value of this command if the cookie name does not collide with any of the other commands.
HTTP::cookie version <name> [version]	Sets or gets the version of the cookie.
HTTP::cookie path <name> [path]	Sets or gets the cookie path.
HTTP::cookie domain <name> [domain]	Sets or gets the cookie domain.
HTTP::cookie ports <name> [portlist]	Sets or gets the cookie port lists for V1 cookies.
HTTP::cookie insert name <name> value <value> [path <path>] [domain <domain>] [version <0 | 1 | 2>]	Adds or replaces a cookie. The default value for the version is **0**.
HTTP::cookie remove <name>	Removes a cookie.
HTTP::cookie sanitize [attribute]+	Removes all but the specified attributes from the cookie.
HTTP::cookie exists <name>	Returns a true value if the cookie exists.
Commands for response messages	
HTTP::cookie names	Returns the names of all the cookies present in the HTTP header.
HTTP::cookie count	Returns the number of cookies present in the HTTP header.
HTTP::cookie [value] <name> [string]	Sets or gets the cookie value of the given name. You can omit the value of this command if the cookie name does not collide with any of the other commands.
HTTP::cookie version <name> [version]	Sets or gets the version of the cookie.
HTTP::cookie path <name> [path]	Sets or gets the cookie path.
HTTP::cookie domain <name> [domain]	Sets/Gets the cookie domain.
HTTP::cookie ports <name> [portlist]	Sets/Gets the cookie port lists for Version 1 cookies.
HTTP::cookie insert name <name> value <value> [path] [domain][version]	Adds or replaces a cookie. The default value for the version is **0**. cookies.
HTTP::cookie remove <name>	Removes a cookie.
HTTP::cookie maxage <name> [seconds]	Sets or gets the max-age. Applies to Version 1 cookies only.
HTTP::cookie expires <name> [seconds] [absolute | relative]	Sets or gets the expires attribute. Applies to Version 0 cookies only. If you specify the **absolute** argument, the **seconds** value represents number of seconds since the UNIX epoch (January 1, 1970). The default number of seconds is **relative**, which is the number of seconds from the current time.
HTTP::cookie comment <name> [comment]	Sets or gets the cookie comment. Applicable only to Version 1 cookies.
HTTP::cookie secure <name> [enable|disable]	Sets or gets the secure attribute. Applicable only to Version 1 cookies.
HTTP::cookie commenturl <name> [commenturl]	Sets or gets the comment URL. Applicable only to Version 1 cookies.
HTTP::cookie discard <name> [enable|disable]	Sets or gets the discard attribute. Applicable only to Version 1 cookies.

Table 13.17 *iRule commands for manipulating cookies in HTTP requests and responses*

iRule Command	Description
HTTP::cookie sanitize [attribute]+	Removes from the cookie all but the attributes you specify.
HTTP::cookie exists <name>	Returns a true value if the cookie exists.
HTTP::cookie encrypt <name> <pass phrase> <data> ["128" | "192" | "256"]	Encrypts the value for the given cookie using a key generated from the pass phrase. The default key length is **128**.
HTTP::cookie decrypt <name> <pass phrase> <data> ["128" | "192" | "256"]	Decrypts the value for the given cookie using a key generated from the pass phrase. The default key length is **128**.

Table 13.17 *iRule commands for manipulating cookies in HTTP requests and responses*

Manipulating SSL headers and content

SSL data manipulation commands that you can use within iRules fall into two categories:

- Commands to extract X509 certificate field values
- Commands to change settings or invoke action

The following two sections describe these commands.

Extracting X509 certificate field values

The iRules feature includes a set of **X509** commands that extract X509 certificate field values. You can use these commands with the **HTTP::header** command when you want to insert, remove, or replace SSL-related headers for HTTP requests. The headers and values that you can query using the X509 iRule commands are listed in Table 13.18.

Header Type	Header Name and 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.

Table 13.18 *SSL header information for use with iRules*

Header Type	Header Name and Format	Description
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 **Unknown 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 13.18 *SSL header information for use with iRules*

Table 13.19 shows the **X509** commands that you can use to extract the value of a specific header. Note that the **X509::cert_fields** command returns a list of all name/value pairs that are suitable for use with the **HTTP::header** command.

iRule Command	Description
X509::version <X509 certificate>	Returns the version number of the X509 certificate (an integer).
X509::serial_number <X509 certificate>	Returns the serial number of the X509 certificate (an integer).
X509::signature_algorithm <X509 certificate>	Returns the signature algorithm of the X509 certificate.
X509::issuer <X509 certificate>	Returns the issuer of the X509 certificate.
X509::not_valid_before <X509 certificate>	Returns the **not-valid-before** date of the X509 certificate.
X509::not_valid_after <X509 certificate>	Returns the **not-valid-after** date of the X509 certificate.
X509::subject <X509 certificate>	Returns the subject of the X509 certificate.
X509::subject_public_key_type <X509 certificate>	Returns the subject's public key type of the X509 certificate. The value can be either **RSA**, **DSA**, or **unknown**.
X509::subject_public_key <X509 certificate>	Returns the subject's public key of the X509 certificate.
X509::subject_public_key_RSA_bits <X509 certificate>	Returns the size in bits of the subject's public RSA key of the X509 certificate. This command is only applicable when the public key type is **RSA**, and generates an error otherwise.
X509::extensions <X509 certificate>	Returns the X509 extensions set on the certificate.
X509::whole <X509 certificate>	Returns the entire X509 certificate in PEM format.
X509::hash <X509 certificate>	Returns the MD5 hash (fingerprint) of the X509 certificate.
X509::verify_cert_error_string <X509 verify error code>	Returns the same result as the OpenSSL function **X509_verify_cert_error_string()**. The **<X509 verify error code>** argument uses the same values as those that the **SSL::verify result** command returns.
X509::cert_fields <X509 certificate> <verify error code> {options}	Returns a list of header name/value pairs suitable for use with the **HTTP::header** command.

Table 13.19 *iRule commands for manipulating SSL query results*

Changing SSL settings or invoking actions

Table 13.20 lists and describes a set of SSL commands that you can use to change SSL settings or invoke actions.

iRule Command	Description
SSL::renegotiate	This command has different results depending on whether the LTM system evaluates the command under a client-side or a server-side context. The command only succeeds if SSL is enabled on the connection; otherwise, the command returns an error. When the system evaluates the command under a client-side context, the system immediately renegotiates a request for the associated client-side connection, using the configuration options for forced SSL renegotiations. This renegotiation enforces any SSL settings changed for the connection including client certificate settings. When the system evaluates the command under a server-side context, the system immediately initiates a renegotiation for the associated server-side connection, using the configuration options for forced SSL renegotiations.
SSL::cert mode <“request” | “require” | “ignore” | “auto”>	This command has different results depending on whether the system evaluates the command under a client-side or server-side context. When the system evaluates the command under a client-side context, the command overrides the client-side SSL connection’s current setting regarding client certificates. When the system evaluates the command under a server-side context, the command overrides the server-side SSL connection’s current setting regarding server certificates. Only the **require** or **ignore** arguments are valid in this instance.
SSL::authenticate <“once” | “always”>	This command is only valid in a client-side context. The command overrides the client-side SSL connection’s current setting regarding authentication frequency.
SSL::authenticate depth <number>	This command has different results depending on whether it is evaluated under a client-side or server-side context. When the system evaluates the command under a client-side context, the command overrides the client-side SSL connection’s current setting regarding maximum certificate chain traversal depth. When the system evaluates the command under a server-side context, the command overrides the server-side SSL connection’s current setting regarding maximum certificate chain traversal depth.
SSL::unclean shutdown <“enable” | “disable”>	This command has different results depending on whether it is evaluated under a client-side or server-side context. When the system evaluates the command under a client-side context, the command overrides the client-side SSL connection’s current setting regarding unclean shutdowns. When the system evaluates the command under a server-side context, the command overrides the server-side SSL connection’s current setting regarding unclean shutdowns.
SSL::verify result <result_code>	Sets the result code for peer certificate verification. The **<result_code>** argument uses the same values as those that the **SSL::verify result** command returns.

Table 13.20 *iRule commands for changing settings or invoking actions*

iRule Command	Description
SSL::handshake hold	Halts any SSL activity upon authentication.
SSL::handshake resume	Resumes any SSL activity that the system previously halted with the **SSL::handshake hold** command.
SSL::session invalidate	Invalidates the current session. Specifically, drops the session from the session cache to prevent reuse of the session.

Table 13.20 *iRule commands for changing settings or invoking actions*

Setting statistical data

You can use certain commands to set the value of counters that you define in a Statistics profile. Table 13.21 lists and describes these commands.

iRule Command	Description
STATS::set <profile> <field> [<value>]	Sets the value of the named field in the named Statistics profile to the specified value. If you do not specify a value, the BIG-IP system sets the value to **0**.
STATS::incr <profile> <field> [<value>]	Increments the value of the named field in the named Statistics profile by the specified value. If you do not specify a value, the BIG-IP system sets the value to **1**. Incrementing a field value by a negative number decrements the field value.
STATS::setmin <profile> <field> [<value>]	Ensures that the value of the named field in the named Statistics profile is at most value.
STATS::setmax<profile> <field> [<value>]	Ensures that the value of the named field in the named Statistics profile is at least value.

Table 13.21 *iRule command for querying Statistics data*

Using utility commands

The LTM system includes a number of utility commands that you can use within iRules. You can use these commands to parse and retrieve content, encode data into ASCII format, verify data integrity, and retrieve information about active pools and pool members.

Parsing and manipulating content

Table 13.22 lists and describes the commands that return a string that you specify. The pages following the table provide detail and examples of the commands.

Command	Description
findstr	Finds a string within another string and returns the string starting at the offset specified from the match.
substr	Finds a string within another string and returns the string starting at the offset specified from the match.
getfield	Splits a string on a character, and returns the string corresponding to the specific field
findclass	Finds the member of a data group that contains the result of the specified expression, and returns that data group member or the portion following the separator, if a separator was provided.
decode_uri	Evaluates the expression, and returns a string with any **%XX** escape sequences decoded as per HTTP escape sequences defined in RFC2396. Note that this command is equivalent to the command **URI::decode**.
domain	Parses and returns up to the specified number of trailing parts of a domain name from the specified expression.

Table 13.22 *Utility commands that parse and manipulate content*

findstr

The **findstr** command finds the string **<search_string>** within **<string>** and returns a sub-string based on the **<skip_count>** and **<terminator>** from the matched location.

Note the following;

- **<terminator>** may be either a character or length.
- If **<skip_count>** is not specified, it defaults to zero.
- If **<terminator>** is not specified, it defaults to the end of the string.
- This command (without **<skip_count>** or **<terminator>**) is equivalent to the following Tcl command: "**string range <string> [string first <string> <search_string>] end**".

The syntax of the **findstr()** command is as follows:

```
findstr <string> <search_string> [<skip_count> [<terminator>]
```

Figure 13.16 shows an example of an iRule using the **findstr** command.

```
when HTTP_REQUEST {
  if { [findstr [HTTP::uri] "type=" 5 "&"] eq "cgi" } {
     pool cgi_servers
  } else {
     pool web_servers
 }
}
```

Figure 13.16 *An iRule using the* ***findstr*** *command*

substr

The **substr** command returns a sub-string **<string>** based on the values of **<skip_count>** and **<terminator>**.

Note the following:

- The **<skip_count>** and **<terminator>** arguments are used in the same way as they are for the **findstr** command.
- This command is equivalent to the Tcl **string range** command except that the value of **<terminator>** may be either a character or a count.

The syntax of the **substr** command is:

```
substr <string> <skip_count> [<terminator>]
```

getfield

The **getfield** command splits a string on a character, and returns the string corresponding to the specific field.

The syntax of the **getfield** command is:

```
getfield <string> <split> <field_number>
```

findclass

The **findclass** command searches a data group list for a member that starts with **<string>** and returns the data-group member string. This is similar to the **matchclass** command, except that the member is not required to be equal; instead, the member is only required to start with the string and the command returns the entire member value.

The syntax of the **findclass** command is:

```
findclass <string> <data group> [(separator)]
```

Note that if a separator is specified, the data group member is split on the separator, and the latter portion (that is, the portion following the separator) is returned.

decode_uri

The **decode_uri** command decodes the string **<string>** using HTTP URI encoding per RFC2616 and returns the result.

The syntax of the **decode_uri** command is:

```
decode_uri <string>
```

domain

The **domain** command parses the string **<string>** as a dotted domain name and return the last **<count>** portions of the domain name.

The syntax of the **domain** command is:

```
domain <string> <count>
```

Encoding data

Some of the commands available for use within iRules allow you to encode data into ASCII format. Table 13.23 lists and describes these commands.

Utility Command	Description
b64encode <string>	Returns a string that is base-64 encoded, or if an error occurs, an empty string.
b64decode <string>	Returns a string that is base-64 decoded, or if an error occurs, an empty string.

Table 13.23 *Utility commands for encoding data*

Ensuring data integrity

Some of the commands available for use within iRules allow you to check the integrity of data. Table 13.24 lists and describes these commands.

Utility Command	Description
crc32 <string>	Returns the **crc32** checksum for the provided string, or if an error occurs, an empty string. Used to ensure data integrity.
md5 <string>	Returns the RSA Data Security, Inc. MD5 Message Digest Algorithm (**md5**) message digest of the provided string, or if an error occurs, an empty string. Used to ensure data integrity.
sha1 <string>	Returns the Secure Hash Algorithm version 1.0 (**SHA1**) message digest of the provided string, or if an error occurs, an empty string. Used to ensure data integrity.

Table 13.24 *Utility commands for ensuring data integrity*

Retrieving pool information

Some of the commands available for use within iRules allow you to retrieve data about pools and pool members. Table 13.25 lists and describes these commands.

Utility Command	Description
active_members <pool name>	Returns the number of active members in the pool.
active_nodes <pool name>	Returns the alias for active pool members (for 4.X compatibility).
member_priority <pool name> member <ip> [<port>]	Returns the priority for pool member **ip:port**.

Table 13.25 *Utility commands for retrieving pool information*

Working with profiles

When you are writing an iRule, you might want that iRule to know the value of a particular profile setting so that it can make a more-informed traffic management decision. Fortunately, the iRules feature includes a command that is specifically designed to read the value of profile settings that you specify within the iRule.

Not only can iRules read the values of profile settings, but they can also override values for certain settings. This means that you can apply configuration values to individual connections that differ from the values the LTM system applies to most connections passing through a virtual server.

Reading profile settings

The iRules feature includes a command called **PROFILE**. When you specify the **PROFILE** command in an iRule and name a profile type and setting, the iRule reads the value of that particular profile setting. To do this, the iRule finds the named profile type that is assigned to the virtual server and reads the value of the setting that you specified in the **PROFILE** command sequence. The iRule can then use this information to manage traffic.

For example, you can specify the command **PROFILE::tcp idle_timeout** within your iRule. The LTM system then finds the TCP profile that is assigned to the virtual server (for example, **my_tcp**) and queries for the value that you assigned to the **Idle Timeout** setting.

Overriding profile settings

Some of the iRule commands for querying and manipulating header and content data have equivalent settings within various profiles. When you use those commands in an iRule, and an event triggers that iRule, the LTM system overrides the values of those profile settings, using the value specified within the iRule instead.

For example, an HTTP profile might specify a certain buffer size to use for compressing HTTP data, but you might want to specify a different buffer size for a particular type of HTTP connection. In this case, you can include the command **HTTP::compress_buffer_size** in your iRule, specifying a different value than the value in the profile.

Enabling session persistence with iRules

Chapter 9, *Enabling Session Persistence*, describes how to enable session persistence by configuring a persistence profile and assigning it to a virtual server. As described in that chapter, the LTM system applies those persistence profile settings to every applicable session that passes through the virtual server. For example, if you have assigned the **msrdp** profile to the virtual server, then the LTM system applies those settings to every incoming Microsoft® Remote Desktop Protocol (RDP) connection.

There are cases, however, when you might want to enable persistence in a more granular way. For example, instead of using the **ssl** persistence profile, which acts on non-terminated SSL traffic only, you might want to persist sessions based on SSL certificate status that you insert into the header of an HTTP request. To do this, you write an iRule using the **HTTP::header** command and then assign the iRule to the virtual server. Whenever the LTM system accepts an SSL request, the iRule inserts the certificate status as a header into the request, and persists the session based on that status.

The LTM system includes a special iRule command, **persist**, for implementing the types of session persistence described in Chapter 9, *Enabling Session Persistence*. You simply type the **persist** command in your iRule, specifying a persistence type.

For example, you can write an iRule that enables persistence for SSL connections when a particular event occurs, basing the persistence on the session ID. Figure 13.17 shows an example of an iRule that you could write to do this:.

```
when CLIENTSSL_HANDSHAKE {
    persist ssl
}
```

Figure 13.17 *Sample iRule for SSL persistence based on session ID*

The following list shows the **persist** command and the persistence-related keywords that you can specify when defining a persistence type within an iRule. For some persistence keywords, you must specify additional arguments.

- **persist cookie [insert | rewrite | passive | hash] [<cookie name>]**
- **persist [dest_addr | sticky] [<network IP address>] [<timeout>]**
- **persist hash**
- **persist msrdp**
- **persist sip**
- **persist [src_addr | simple] [<network IP address>] [<timeout>]**
- **persist ssl**
- **persist [uie | universal] <string> [<timeout>]**
- **persist none**

You can use the **persist none**, **hash**, **srcaddr**, **destaddr**, and **uie** commands in any circumstance, even if a corresponding persistence profile is not configured and assigned to the virtual server. However, the **persist ssl, cookie, msrdp,** and **sip** commands require that you assign a corresponding persistence profile to the virtual server. Attempts to use these commands without a corresponding profile result in a run-time iRule error. For information on assigning a persistence profile to a virtual server, see Chapter 2, *Configuring Virtual Servers*.

Creating, managing, and using data groups

Data groups are useful when writing iRules. A ***data group*** is simply a group of related elements, such as a set of IP addresses for AOL clients. When you specify a data group along with the **matchclass** command or the **contains** operator, you eliminate the need to list multiple values as arguments in an iRule expression

To understand the usefulness of data groups, it is helpful to first understand the **matchclass** command and the **contains** operator.

Using the matchclass command

The LTM system includes an iRule command called **matchclass**, which you can use to select a pool based on whether the command being used in the iRule represents a member of a specific data group. When you use the **matchclass** command, the LTM system knows that the string following the identifier is the name of a data group.

For example, using the **matchclass** command, you can cause the LTM system to load balance all incoming AOL connections to the pool **aol_pool,** if the value of the **IP::remote_addr** command is a member of the data group AOL. Figure 13.18 shows this type of iRule. In this case, the **matchclass** command simply indicates that the object named **aol** is a collection of values (that is, a data group).

```
when CLIENT_ACCEPTED {
  if { [matchclass [IP::remote_addr] equals $::aol] } {
     pool aol_pool
  } else {
     pool all_pool
 }
}
```

Figure 13.18 *An iRule based on the* ***matchclass*** *command*

Note that an expression such as **[IP::remote__addr] equals matchclass $::aol** is true if the expression is true with at least one specific value in the data group.

Creating data groups

When using the **matchclass** command within an iRule, you can specify any of three types of data groups:

- Addresses data group - A collection of IP addresses
- String data group - A collection of strings, such as ***.jpg**
- Integer data group - A collection of numeric values.

The following sections describe these data group types.

Note

The size of a data group is limited by system resources only.

Address data groups

There are two types of IP address data groups, network IP address and host IP address.

The following procedure creates a network or host address data group:

To create an address data group

1. On the Main tab, expand **Local Traffic**.
2. Click **iRules**.
   The iRules screen opens.
3. On the menu bar, click **Data Group List**.
4. In the upper right corner of the screen, click **Create**.
5. In the **Name** box, type a unique name for the data group, such as **my_address_group**.
6. In the **Type** box, select **Address**.
   The screen expands to show more settings.
7. In the Records section, select the **Type** you want, **Host** or **Network**.
8. In the Address box, type the first IP address for the data group. If you are creating a network data group, also enter a network mask in the **Mask** box.
9. Click **Add**.
   The entry appears in the **Address Records** box.
10. Repeat steps 7 and 8 until you have entered all IP addresses.
11. Click **Finished**.

String data groups

A string data group contains a list of strings, such as ***.jpg** or ***.gif**. The following procedure creates a string data group.

Note that this example shows the use of escape characters for the quotation marks.

To create a string data group

1. On the Main tab, expand **Local Traffic**.
2. Click **iRules**.
   The iRules screen opens.

3. On the menu bar, click **Data Group List**.
4. In the upper right corner of the screen, click **Create**.
5. In the **Name** box, type a unique name for the data group, such as **my__images**.
6. In the **Type** box, select **String**.
   The screen expands to show the string-specific settings.
7. In the **String** box, type the first string for the data group.
8. Click **Add**.
   The entry appears in the **String Records** box.
9. Repeat steps 6 and 7 until you have entered all strings.
10. Click **Finished**.

Integer data groups

An integer data group contains a list of integers. The following procedure describes how to create an integer data group.

To create an integer data group

1. On the Main tab, expand **Local Traffic**.
2. Click **iRules**.
   The iRules screen opens.
3. On the menu bar, click **Data Group List**.
4. In the upper right corner of the screen, click **Create**.
5. In the **Name** box, type a unique name for the data group, such as **my__integer_group**.
6. In the **Type** box, select **Integer**.
   The screen expands to display the Records section.
7. In the **Integer** box, type the first integer for the data group.
8. Click **Add**.
   The entry appears in the **Integer Records** box.
9. Repeat steps 6 and 7 until you have added all integers.
10. Click **Finished**.

Storage options

The LTM system allows you to store data groups in two ways, either in-line or externally.

In-line storage

When you create data groups, the LTM system automatically saves them in their entirety in the **bigip.conf** file. This type of storage is known as ***in-line storage***.

When any data in the data group needs to be updated, the entire data group must be reloaded. In general, in-line storage uses additional system resources due to extensive searching requirements on large data groups. Also, in-line storage requires you to reload entire data groups when incrementally updating data. For these reasons, the LTM system offers you the ability to store you data groups externally, that is, outside of the **bigip.conf** file.

External storage

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

Creating external data groups 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 an iRule uses a large data group to direct traffic to a pool.

To store data groups externally

1. On the Main tab, expand **Local Traffic**.
2. Click i**Rules**.
   The iRules screen opens.
3. On the menu bar, click **Data Group List**.
4. In the upper right corner of the screen, click **Create**.
5. In the **Name** box, type the name of the existing data group that you want to store in an external location.
6. In the **Type** box, select **(External File)**.
   The screen expands to display the Records section.
7. Specify a storage location:
   - If you do not want to store your data group in the default external location (**/config**), use the **Path / Filename** box to specify a path name and file name for the external location, for example, **/home/my_address_group**.
     This file name should match the name that you assigned to the data group itself.
   - If you want to store your data group in the default external location (**/config**), leave the **Path / Filename** box empty.

8. In the **File Contents** box, select the file type that pertains to the data group (**Address**, **String**, or **Integer**).
9. Click **Finished**.

The LTM system stores the data in an external data group file in comma-separated lists, and the formats of any data values, such as IP addresses, match the formats used in the **bigip.conf** file. Figure 13.19 shows the contents of the data group file **/home/ip2.data group**.

```
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 13.19 *An example of an external data group file*

Displaying data group properties

Using the Configuration utility, you can display the properties of an existing data group.

To display the properties of an data group

1. On the Main tab, expand **Local Traffic**.
2. Click **iRules**.
   The iRules screen opens.
3. On the menu bar, click **Data Group List**.
4. Click the name of a data group.
   This displays the properties of that data group.

Managing data group members

Using the Configuration utility, you can add members to or delete members from an existing data group.

To add members to a data group

1. On the Main tab, expand **Local Traffic**.
2. Click **iRules**.
   The iRules screen opens.
3. On the menu bar, click **Data Group List**.
4. Click the name of a data group.
   This displays the properties of that data group.
5. Locate the records box and type an address, string, or integer in the appropriate box.

6. Click **Add**.
7. At the bottom of the screen, click **Update**.

To delete members from a data group

1. On the Main tab, expand **Local Traffic**.
2. Click **iRules**.
   The iRules screen opens.
3. On the menu bar, click **Data Group List**.
4. Click the name of a data group.
   This displays the properties of that data group.
5. Check the **Select** box for the member you want to delete.
6. Click the **Delete** button, and when the confirmation message appears, click **Delete** again.

Note

*When you synchronize a BIG-IP system redundant-system configuration, the LTM system only includes data groups that reside in the **bigip.conf** file (that is, in-line stored data groups). If you want the synchronization to include externally-stored data groups, you must add an associated entry for each data group into the **bigip.conf** file.*

A

Additional Monitor Considerations

- Implementing monitors for Dynamic Ratio load balancing
- Implementing an MSSQL monitor

Implementing monitors for 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 for these types of servers, the BIG-IP® local traffic management (LTM) system provides a special monitor plug-in file and a health or performance monitor for each type of server. The exception is a server equipped with an SNMP agent. In this case, the LTM system provides the monitor only; no special plug-in file is required for a server running an SNMP agent.

You must install the monitor plug-in on each server to be monitored, and you must create a performance monitor that resides on the LTM system. Once you have created a monitor, the monitor communicates directly with the server plug-in. For each server type, Table A.1 shows the required monitor plug-in and the corresponding performance monitor types.

Server Type	Monitor plug-in	Monitor Type
RealServer Windows server	**F5RealMon.dll**	Real Server
RealServer UNIX server	**f5realmon.so**	Real Server
Windows server with WMI	**F5Isapi.dll**	WMI
Windows 2000 Server server	SNMP agent	SNMP DCA and SNMP DCA Base
UNIX server	UC Davis SNMP agent	SNMP DCA and SNMP DCA Base

Table A.1 *Monitor plug-ins and corresponding monitor templates*

Implementing a Real Server monitor

For RealSystem Server systems, the LTM system provides a monitor plug-in that gathers the necessary metrics when you have installed the plug-in on the RealSystem Server system. Configuring a RealSystem Server for Dynamic Ratio load balancing consists of four tasks:

- Installing the monitor plug-in on the RealSystem server
- Configuring a Real Server monitor on the LTM system
- Associating the monitor with the server to gather the metrics
- Creating or modifying the server pool to use Dynamic Ratio load balancing

To install the monitor plug-in on a RealSystem Server system (Windows version)

1. Download the monitor plug-in **F5RealServerPlugin.dll** from the LTM system.
   The plug-in is located in the folder **/usr/local/www/docs/agents**.
2. Copy **F5RealServerPlugin.dll** to the RealServer **plug-ins** directory. (For example, **C:\Program Files\RealServer\plug-ins**.)
3. If the RealSystem Server process is running, restart it.

To install and compile a Linux or UNIX RealSystem Server monitor plug-in

1. Using the **.iso** image, burn a CD-ROM of the BIG-IP system software.
2. On the CD, navigate to the directory **/downloads/rsplug-ins**.
3. Copy the file **F5RealMon.src.tar.gz** to the directory **/var/tmp** on the BIG-IP system.
4. On the BIG-IP system, change to the directory **/var/tmp**:

   ```
   cd /var/tmp
   ```

5. Use the UNIX **tar** command to uncompress the file **F5RealMon.src.tar.gz**:

   ```
   tar -xvzf F5RealMon.src.tar
   ```

6. Change to the **F5RealMon.src** directory:

   ```
   cd F5RealMon.src
   ```

7. Type the **ls** command to view the directory contents.
8. To compile the source, use the instructions in the file **build_unix_note**.
9. Start RealSystem Server.

Once the plug-in is installed and compiled, you must configure a Real Server monitor, associate the configured monitor with the pool member (a RealSystem Server server), and set the load balancing method to Dynamic Ratio:

- To configure a Real Server monitor, see Chapter 10, *Configuring Monitors*.
- To associate the performance monitor with the pool member, see Chapter 4, *Configuring Load Balancing Pools*.
- To set the load balancing method on the pool to the Dynamic Ratio method, see Chapter 4, *Configuring Load Balancing Pools*.

Implementing a WMI monitor

For Windows running Windows Management Instrumentation (WMI), the LTM 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 monitor on the LTM system
- Associating the monitor with the server to gather the metrics
- Creating or modifying the server pool to use the Dynamic Ratio load balancing method

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

1. Download the **Data Gathering Agent (F5Isapi.dll)** from the LTM system.
   The plug-in 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 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.

Once you have installed the plug-in, you must configure a WMI monitor, associate the configured monitor with the pool member, and set the load balancing to Dynamic Ratio:

- To configure a WMI monitor, see Chapter 10, *Configuring Monitors*.
- To associate the custom monitor with the pool member, see Chapter 4, *Configuring Load Balancing Pools*.
- To set the load balancing method on the pool to the Dynamic Ratio method, see Chapter 4, *Configuring Load Balancing Pools*.

Implementing an SNMP DCA or SNMP DCA Base monitor

The LTM 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.

The LTM system provides two monitor types that you can use to create a performance monitor for a server that uses an SNMP agent. These two monitor types are:

- **SNMP DCA**
  Use this monitor 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.
- **SBMP DCA Base**
  Use this monitor when you want to use default values or specify values for metrics other than CPU, memory, and disk usage. When using this monitor, values for CPU, memory, and disk metrics are omitted.

Configuring a server to use its SNMP agent for Dynamic Ratio load balancing consists of three tasks: configuring an SNMP DCA or SNMP DCA Base monitor, associating the monitor with the applicable pool member, and setting the load balancing method on the pool to the Dynamic Ratio method. For more information, see the following chapters or sections of this guide:

- To configure an SNMP DCA or SNMP DCA Base monitor, see Chapter 10, *Configuring Monitors*.
- To associate a monitor with the pool, see Chapter 4, *Configuring Load Balancing Pools*.
- To set the load balancing method on the pool to the Dynamic Ratio method, see Chapter 4, *Configuring Load Balancing Pools*.

Implementing an MSSQL monitor

Before you can use an MSSQL type of monitor, you must download a set of JDBC Java™ Archive (JAR) files from the Microsoft web site and install them on the BIG-IP system.

To download and install Microsoft JDBC files

1. From an Internet browser, go to **www.microsoft.com**.
2. On the left panel, under Resources, click **Download**.
   A list appears.
3. Select **SQL Server**.
4. In the **Keyword** field, type **JDBC Driver**.
5. Click **Go**.
   This displays a list of options.
6. Click **Microsoft SQL Server 2000 Driver for JDBC**.
7. Verify that you are downloading the UNIX **.tar** file, and not a Windows package. If not, go back to the previous screen.
8. On the right side of the screen, click **Download**.
9. When queried, select **Save the file to disk**.
10. Create a Linux directory, move the **.tar** file to that directory, and uncompress the file, using this command:

    ```
    tar -xvf mssqlserver.tar
    ```

    This command extracts four files: **EULA.txt**, **install.ksh**, **msjdbc.tar**, and **read.me**.
11. Install these files by typing **install.ksh** at the command line prompt. This untars the **msjdbc.tar** file, which creates several subdirectories.
12. Locate the **lib** subdirectory.
    This directory contains the three JAR files listed previously.
13. Copy the three JAR files to the BIG-IP directory **/usr/bin/monitors/builtins/**.
    You can now recursively remove the LINUX directory that you created in step 10.

After installing the JAR files, we recommend that you either reboot the BIG-IP system, or run the following command:

```
/usr/bin/monitors/builtins/DB_monitor cmd quit
```

Rebooting the system or running this command causes the BIG-IP system to recognize the newly-installed JAR files the next time that you run an MSSQL monitor.

B

Disabled Tcl Commands

- Disabled Tcl commands

Disabled Tcl commands

When you use the iRules™ feature of a BIG-IP® local traffic management (LTM) system to create scripts for managing local network traffic, you use the industry-standard Tool Command Language (Tcl). However, a subset of Tcl commands are disabled on the LTM system. You should therefore avoid using these commands when writing iRules.

The Tcl commands that are disabled on an LTM system are:

- **after**
- **auto_execok**
- **auto_import**
- **auto_load**
- **auto_mkindex**
- **auto_mkindex_old**
- **auto_qualify**
- **auto_reset**
- **bgerror**
- **cd**
- **close**
- **eof**
- **exec**
- **exit**
- **fblocked**
- **fconfigure**
- **fcopy**
- **file**
- **fileevent**
- **filename**
- **flush**
- **gets**
- **glob**
- **http**
- **interp**
- **load**
- **memory**
- **namespace**
- **open**
- **package**
- **pid**
- **pkg::create**

- **pkg_mkIndex**
- **proc**
- **pwd**
- **rename**
- **seek**
- **socket**
- **source**
- **tcl_findLibrary**
- **tell**
- **unknown**
- **update**
- **uplevel**
- **upvar**
- **vwait**

Glossary

active unit

In a redundant system, the active unit is the 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. See also *redundant system*.

archive

An archive is a backup copy of the BIG-IP system configuration data. This archive is in the form of a user configuration set, or UCS. See also *user configuration set (UCS)*.

ARP (Address Resolution Protocol)

ARP is an industry-standard protocol that determines a host's Media Access Control (MAC) address based on its IP address.

authentication

Authentication is the process of verifying a user's identity when the user is attempting to log on to a system.

authentication iRule

An authentication iRule is a system-supplied or user-created iRule that is necessary for implementing a PAM authentication module on the LTM system. See also *iRule*, *PAM (Pluggable Authentication Module)*.

authentication module

An authentication module is a PAM module that you create to perform authentication or authorization of client traffic. See also *PAM (Pluggable Authentication Module)*.

authentication profile

An authentication profile is a configuration tool that you use to implement a PAM authentication module. Types of authentication modules that you can implement with an authentication profile are: LDAP, RADIUS, TACACS+, SSL Client Certificate LDAP, and OCSP. See also *PAM (Pluggable Authentication Module)*.

authorization

Authorization is the process of identifying the level of access that a logged-on user has been granted to system resources.

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 the Domain Name System (DNS). BIND provides a system for matching domain names to IP addresses. For more information, refer to **http://www.isc.org/products/BIND**.

BPDU (bridge protocol data unit)

A BPDU is a special packet that a spanning tree protocol sends between layer 2 devices to determine redundant paths, and provide loop resolution. See also *STP (Spanning Tree Protocol)*, *RSTP (Rapid Spanning Tree Protocol)*, and *MSTP (Multiple Spanning Tree Protocol)*.

bursting

Bursting is an aspect of rate shaping and occurs when the rate of traffic flow exceeds the base rate defined.

certificate

A certificate is an online credential signed by a trusted certificate authority and used for SSL network traffic as a method of authentication.

certificate authority (CA)

A certificate authority is an external, trusted organization that issues a signed digital certificate to a requesting computer system for use as a credential to obtain authentication for SSL network traffic.

certificate revocation list (CRL)

A certificate revocation list is a list that an authenticating system checks to see if the SSL certificate that the requesting system presents for authentication has been revoked.

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.

chunking

See *HTTP chunking*.

cipher

A cipher is an encryption/decryption algorithm that computer systems use when transmitting data using the SSL protocol.

client-side SSL profile

A client-side SSL profile is an SSL profile that controls the behavior of SSL traffic going from a client system to the LTM system.

clone pool

This feature causes a pool to replicate all traffic coming into it and send that traffic to a duplicate pool.

configuration object

A configuration object is a user-created object that the LTM system uses to implement a PAM authentication module. There is one type of configuration object for each type of authentication module that you create. See also *PAM (Pluggable Authentication Module)*.

configuration synchronization

Configuration synchronization is the task of duplicating a BIG-IP system's configuration data onto its peer unit in a redundant system.

Configuration utility

The Configuration utility is the browser-based application that you use to configure the LTM system.

connection persistence

Connection persistence is an optimization technique whereby a network connection is intentionally kept open for the purpose of reducing handshaking.

connection pooling

Connection pooling is an optimization feature that pools server-side connections for re-use by other client requests. Connection pooling reduces the number of new connections that must be opened for server-side client requests.

content switching

Content switching is the ability to select a pool based on data contained within a packet.

cookie persistence

Cookie persistence is a mode of persistence where the LTM system stores persistent connection information in a cookie.

custom profile

A custom profile is a profile that you create. A custom profile can inherit its default settings from a parent profile that you specify. See also *parent profile*.

data group

A data group is a group of related elements, such as a set of IP addresses for AOL clients. When you specify a data group along with the **matchclass** command or the **contains** operator, you eliminate the need to list multiple values as arguments in an iRule expression.

default profile

A default profile is a profile that the LTM system supplies with default setting values. You can use a default profile as is, or you can modify it. You can also specify it as a parent profile when you create a custom profile. You cannot create or delete a default profile. See also *profile*, *custom profile*.

default route

A default route is the route that the system uses when no other route specified in the routing table matches the destination address or network of the packet to be routed.

default VLAN

The LTM 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.

destination address affinity persistence

Also known as sticky persistence, destination address affinity persistence supports TCP and UDP protocols, and directs session requests to the same server based solely on the destination IP address of a packet.

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.siterequest.com/index.html**, the domain name is **siterequest.com**.

Dynamic Ratio load balancing method

Dynamic Ratio mode is like Ratio mode (see *Ratio method*), except that ratio weights are based on continuous monitoring of the servers and are therefore continually changing. Dynamic Ratio load balancing can be implemented on RealNetworks® RealServer platforms, on Microsoft® 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 route

A dynamic route is a route that an advanced routing protocol such as RIP adds dynamically to a routing table. See also *static route*.

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 an LTM 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 an LTM system. See also *health check*.

external authentication

External authentication refers to the process of using a remote server to store data for the purpose of authenticating users or applications attempting to access the LTM 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.

failback

Failback is the process whereby an active unit relinquishes processing to a previously-failed unit that is now available.

failover

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.

failover cable

The fail-over cable directly connects the two units together in a redundant system.

Fastest method

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.

forwarding virtual server

A forwarding virtual server is a virtual server that has no pool members to load balance. The virtual server simply forwards the packet directly to the destination IP address specified in the client request. See also *virtual server*.

gateway pool

A gateway pool is a pool of routers that you can create to forward traffic. After creating a gateway pool, you can specify the pool as a gateway, within a TMM routing table entry.

hash persistence

Hash persistence allows you to create a persistence hash based on an existing iRule.

health check

A health check is an LTM 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*.

host virtual server

A host virtual server is a virtual server that represents a specific site, such as an Internet web site or an FTP site, and it load balances traffic targeted to content servers that are members of a pool.

HTTP chunking

HTTP chunking refers to the HTTP/ 1.1 feature known as chunked encoding, which allows HTTP messages to be broken up into several parts. Chunking is most often used by servers when sending responses.

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**.

HTTP transformation

When the LTM system performs an HTTP transformation, the system manipulates the **Connection** header of a server-side HTTP request, to ensure that the connection stays open. See also *connection persistence*.

ICMP (Internet Control Message Protocol)

ICMP is an Internet communications protocol used to determine information about routes to destination addresses.

i-mode

i-mode® is a service created by NTT DoCoMo, Inc., that allows mobile phone users access to the Internet.

interface

The physical port on a BIG-IP system is called an interface.

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.

iRule

An iRule is a user-written script that controls the behavior of a connection passing through the LTM system. iRules™ are an F5 Networks feature and are frequently used to direct certain connections to a non-default load balancing pool. However, iRules can perform other tasks, such as implementing secure network address translation and enabling session persistence.

iSNAT (intelligent SNAT)

An iSNAT is the mapping of one or more original client IP addresses to a translation address from within an iRule. Before writing an iRule to create an iSNAT, you must create a SNAT pool. See also *SNAT pool*.

JAR file

A JAR file is a file in Java™ Archive (JAR) file format that enables you to bundle multiple files into a single archive file. Typically, a JAR file contains the class files and auxiliary resources associated with applets and applications.

JDBC

JDBC is a Java™ technology. It is an application programming interface that provides database management system (DBMS) connectivity across a wide range of SQL databases, as well as access to other tabular data sources, such as spreadsheets or flat files.

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.

Link Aggregation Control Protocol (LACP)

LACP is an industry-standard protocol that aggregates links in a trunk, to increase bandwidth and provide for link failover.

last hop

A last hop is the final hop a connection takes 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.

layer 1 through layer 7

Layers 1 through 7 refer to the seven layers of the Open System Interconnection (OSI) model. Thus, layer 2 represents the data-link layer, layer 3 represents the IP layer, and layer 4 represents the transport layer (TCP and UDP). Layer 7 represents the application layer, handling traffic such as HTTP and SSL.

layer 2 forwarding table

A layer 2 forwarding table correlates MAC addresses of network devices to the BIG-IP system interfaces through which those devices are accessible. On a BIG-IP system, each VLAN has its own layer 2 forwarding table.

LDAP (Lightweight Directory Access Protocol)

LDAP is an Internet protocol that email programs use to look up contact information from a server.

LDAP authentication module

An LDAP authentication module is a user-created module that you implement on an LTM system to authenticate client traffic using a remote LDAP server.

LDAP client certificate SSL authentication module

An LDAP client certificate SSL authentication module is a user-created module that you implement on an LTM system to authorize client traffic using SSL client credentials and a remote LDAP server.

Least Connections method

Least Connections mode is a dynamic load balancing method that bases connection distribution on which server currently manages the fewest open connections.

link aggregation

Link aggregation is the process of combining multiple links in order to function as though it were a single link with higher bandwidth. Link aggregation occurs when you create a trunk. See also *trunk* and *Link Aggregation Control Protocol (LACP)*.

load balancing method

A particular method of determining how to distribute connections across a load balancing pool.

load balancing pool

See *pool*.

load balancing virtual server

A load balancing virtual server is a virtual server that directs client traffic to a load balancing pool. This is the most basic type of virtual server. See also *virtual server*.

local traffic management (LTM)

Local traffic management (LTM) is the process of managing network traffic that comes into or goes out of a local area network (LAN), including an intranet.

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.

management interface

The management interface is a special port on the BIG-IP system, used for managing administrative traffic. Named **MGMT**, the management interface does not forward user application traffic, such as traffic slated for load balancing. See also *TMM switch interface*.

management route

A management route is a route that forwards traffic through the special management (**MGMT**) interface.

MCPD service

The Master Control Program Daemon (MCPD) service manages the configuration data on a BIG-IP system.

MindTerm SSH

MindTerm SSH is the third-party application on 3-DNS Controllers that uses SSH for secure remote communications. SSH encrypts all network traffic (including passwords) to effectively eliminate eavesdropping, connection hijacking, and other network-level attacks. SSH also provides secure tunneling capabilities and a variety of authentication methods.

minimum active members

The minimum active members is the number of members that must be active in a priority group in order for the LTM 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).

monitor

The LTM 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 association

A monitor association is an association that a user makes between a health or performance monitor and a pool, pool member, or node.

monitor instance

You create a monitor instance when a health monitor is associated with a pool member or node. It is the monitor instance that actually performs the health check, not the monitor.

monitor template

A monitor template is an internal mechanism that the LTM system uses to provide default values for a custom monitor when no pre-configured monitor exists.

MSRDP persistence

MSRDP persistence tracks sessions between clients and servers running the Microsoft® Remote Desktop Protocol (RDP) service.

MSTP (Multiple Spanning Tree Protocol)

Defined by IEEE, MSTP is an enhanced spanning tree protocol. Unlike STP and RSTP, MSTP is VLAN-aware and therefore incorporates the concept of MSTP regions. See also *STP (Spanning Tree Protocol*) and *RSTP (Rapid Spanning Tree Protocol).*

MSTP region

An ***MSTP region*** is a group of layer 2 devices that have identical values for certain configuration settings. When devices constitute a region, the spanning tree algorithm takes VLANs into account when blocking and unblocking redundant paths.

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.

NAT (Network Address Translation)

A NAT is an alias IP address that identifies a specific node managed by the LTM system to the external network.

network virtual server

A network virtual server is a virtual server whose IP address has no bits set in the host portion of the IP address (that is, the host portion of its IP address is **0**). There are two kinds of network virtual servers: those that direct client traffic based on a range of destination IP addresses, and those that direct client traffic based on specific destination IP addresses that the LTM system does not recognize.

node

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 LTM system uses to verify the status of multiple nodes. When the LTM system uses a node alias to check node status, it pings the node alias. If the LTM system receives a response to the ping, it marks all nodes associated with the node alias as **up**. If the LTM system does not receive a response to the ping, 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 LTM system uses the node ping and health check features to determine node status.

Observed method

Observed mode is a dynamic load balancing method 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.

OCSP (Online Certificate Status Protocol)

OCSP is a protocol that authenticating systems can use to check on the revocation status of digitally-signed SSL certificates. The use of OCSP is an alternative to the use of a certificate revocation list (CRL). See also *certificate revocation list (CRL)*.

OCSP authentication module

An OCSP authentication module is a user-created module that you implement on an LTM system to authenticate client traffic using a remote OCSP responder. The purpose of an OCSP authentication module is to check on the revocation status of a client SSL certificate.

OCSP responder

An OCSP responder is an external server used for communicating SSL certificate revocation status to an authentication server such as the LTM system.

OCSP responder object

A responder object is a software application on the LTM system that communicates with an OCSP responder, for the purpose of checking revocation status of a client or server SSL certificate.

OneConnect™

The F5 Networks OneConnect™ feature optimizes the use of network connections by keeping server-side connections open and pooling them for re-use.

packet rate

The packet rate is the number of data packets per second processed by a server.

PAM (Pluggable Authentication Module)

A PAM module is a software module that a server application uses to authenticate client traffic. The modular design of a PAM module allows an organization to add, replace, or remove that authentication mechanism from a server application with minimal impact to that application. An example of a PAM module is an application that uses a remote Lightweight Directory Access Protocol (LDAP) server to authenticate client traffic. See also *LDAP (Lightweight Directory Access Protocol)*.

parent profile

A parent profile is a profile that can propagate its values to another profile. A parent profile can be either a default profile or a custom profile. See also *profile*.

performance monitor

A performance monitor gathers statistics and checks the state of a target device.

persistence

See *connection persistence* or *session persistence*.

persistence profile

A persistence profile is a configuration tool for implementing a specific type of session persistence. An example of a persistence profile type is a cookie persistence profile.

pipelining

Pipelining is a feature of HTTP/1.1 that allows clients to make requests even when prior requests have not yet received a response from the server.

pool

A pool is composed of a group of network devices (called members). The LTM 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 member

A pool member is a server that is a member of a load balancing 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 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.

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*.

pre-configured monitor

A pre-configured monitor is a system-supplied health or performance monitor. You can use a pre-configured monitor as is, but you cannot modify or delete one. See also *monitor*.

Predictive method

Predictive mode is a dynamic load balancing method 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 method also ranks server performance over time, and passes connections to servers which exhibit an improvement in performance rather than a decline.

profile

A profile is a configuration tool containing settings for defining the behavior of network traffic. The LTM system contains profiles for managing FastL4, HTTP, TCP, FTP, SSL, and RTSP traffic, as well as for implementing persistence and application authentication.

profile setting

A profile setting is a configuration attribute within a profile that has a value associated with it. You can configure a profile setting to customize the way that the LTM system manages a type of traffic.

profile type

A profile type is a category of profile that you use for a specific purpose. An example of a profile type is an HTTP profile, which you configure to manage HTTP network traffic.

protocol profile

A protocol profile is a profile that you create for controlling the behavior of FastL4, TCP, UDP, OneConnect, and RTSP traffic.

Quality of Service (QoS) level

The Quality of Service (QoS) level is a means by which network equipment can identify and treat traffic differently based on an identifier. Essentially, the QoS level specified in a packet enforces a throughput policy for that packet.

RADIUS (Remote Authentication Dial-in User Service)

RADIUS is a service that performs remote user authentication and accounting. Its primary use is for Internet Service Providers, though it can also be used on any network that needs a centralized authentication and/or accounting service for its workstations.

RADIUS authentication module

A RADIUS authentication module is a user-created module that you implement on an LTM system to authenticate client traffic using a remote RADIUS server.

RAM cache

A RAM cache is a cache of HTTP objects stored in the BIG-IP system's RAM that subsequent connections reuse to reduce the amount of load on the back-end servers.

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 shaping*.

rate shaping

Rate shaping is a type of extended IP filter. Rate shaping uses the same IP filter method but applies a rate class, which determines the volume of network traffic allowed. See also *rate class*.

ratio

A ratio is a parameter that assigns a weight to a virtual server for load balancing purposes.

Ratio method

The Ratio load balancing method distributes connections across an array of virtual servers in proportion to the ratio weights assigned to each individual virtual server.

Real-Time Stream Protocol (RTSP)

See *RTSP*.

receive expression

A receive expression is the text string that the LTM 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 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.

referenence link

A reference link is the lowest-numbered interface in a trunk and is used for link aggregation.

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.

responder object

See *OCSP responder object*.

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.

router

A router is a layer 3 networking device. If no VLANs are defined on the network, a router defines a broadcast domain.

RSTP (Rapid Spanning Tree Protocol)

Defined by IEEE, RSTP is an enhanced version of STP (Spanning Tree Protocol). RSTP provides faster spanning tree performance compared to STP. See also *STP* (*Spanning Tree Protocol)* and *MSTP (Multiple Spanning Tree Protocol)*.

RTSP

RTSP (Real-Time Streaming Protocol) establishes and controls one or more time-synchronized streams of continuous media such as audio or video.

secure network address translation (SNAT)

See *SNAT (secure network address translation)*. See also *iSNAT*.

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 LTM system sends to the web server during an extended content verification (ECV) health check.

server-side SSL profile

A server-side SSL profile is an SSL profile that controls SSL traffic going between an LTM system and a destination server system.

service

Service refers to services such as TCP, UDP, HTTP, and FTP.

services profile

A services profile is a configuration tool on the LTM system for managing either HTTP or FTP network traffic.

session persistence

A series of related connections received from the same client, having the same session ID. When persistence is enabled, an LTM system sends all connections having the same session ID to the same node, instead of load balancing the connections. Session persistence is not to be confused with *connection persistence*.

Setup utility

The Setup utility walks you through the initial system configuration process. You can run the Setup utility from the Configuration utility start page.

simple persistence

See *source address affinity persistence*.

SIP persistence

SIP persistence is a type of persistence used for servers that receive Session Initiation Protocol (SIP) messages sent through UDP. SIP is a protocol that enables real-time messaging, voice, data, and video.

SNAT (Secure Network Address Translation)

A SNAT is a feature you can configure on the LTM 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. See also *Standard SNAT* and *iSNAT*.

SNAT pool

A SNAT pool is a pool of translation addresses that you can map to one or more original IP addresses. Translation addresses in a SNAT pool are not self-IP addresses.

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 address affinity persistence

Also known as simple persistence, source address affinity persistence supports TCP and UDP protocols, and directs session requests to the same server based solely on the source IP address of a packet.

source processing

Source processing means that the interface rewrites the source of an incoming packet.

spanning tree

A spanning tree is a logical tree structure of layer 2 devices on a network, created by a spanning tree protocol algorithm and used for resolving network loops.

spanning tree instance

A spanning tree instance is a specific, named spanning tree that a spanning tree protocol creates. See also *spanning tree protocols*.

spanning tree protocols

Spanning tree protocols are the IEEE-specified set of protocols known as Spanning Tree Protocol (STP), Rapid Spanning Tree Protocol (RSTP), and Multiple Spanning Tree Protocol (MSTP). The BIG-IP system includes support for all of these protocols. See also *STP (Spanning Tree Protocol)*, *RSTP (Rapid Spanning Tree Protocol)*, and *MSTP (Multiple Spanning Tree Protocol)*.

SSH

SSH is a protocol for secure remote login and other secure network services over a non-secure network.

SSL (Secure Sockets Layer)

SSL is a network communications protocol that uses public-key technology as a way to transmit data in a secure manner.

SSL persistence

SSL persistence is a type of persistence that tracks non-terminated SSL sessions, using the SSL session ID.

SSL profile

An SSL profile is a configuration tool that you use to terminate and initiate SSL connections from clients and servers.

standard SNAT

A standard SNAT is a SNAT that you implement by using the SNAT screens of the Configuration utility. See also *SNAT* and *iSNAT*.

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.

state mirroring

State mirroring is a feature on the LTM system that preserves connection and persistence information in a redundant system.

static route

A static route is a route that you must explicitly configure on a layer 3 device in its routing table. See also *dynamic route*.

static self IP address

A static self IP address is a self IP address that is not shared between two units of a redundant system.

sticky persistence

See *destination address affinity persistence*.

STP (Spanning Tree Protocol)

Defined by IEEE, STP is a protocol that provides loop resolution in configurations where one or more external switches are connected in parallel with the BIG-IP system. See also *RSTP* (*Rapid Spanning Tree Protocol)* and *MSTP* (*Multiple Spanning Tree Protocol)*.

subdomain

A subdomain is a sub-section of a higher level domain. For example, **.com** is a high level domain, and **F5.com** is a subdomain within the **.com** domain.

TACACS (Terminal Access Controller Access Control System)

TACACS is an older authentication protocol common to UNIX systems. TACACS allows a remote access server to forward a user's login password to an authentication server.

TACACS+

TACACS+ is an authentication mechanism designed as a replacement for the older TACACS protocol. There is little similarity between the two protocols, however, and they are therefore not compatible.

TACACS+ authentication module

A TACACS+ authentication module is a user-created module that you implement on an LTM system to authenticate client traffic using a remote TACACS+ server.

tagged interface

A tagged interface is an interface that you assign to a VLAN in a way that causes the system to add a VLAN tag into the header of any frame passing through that interface. Tagged interfaces are used when you want to assign a single interface to multiple VLANs. See also *VLAN (virtual local area network)*.

Tcl

Tcl (Tools Command Language) is an industry-standard scripting language. On the LTM system, users use Tcl to write iRules™.

TMM (Traffic Management Microkernel) service

The TMM service is the process running on the BIG-IP system that performs most traffic management for the product.

TMM switch interface

A TMM switch interface is an interface that the BIG-IP system uses to forward user application traffic such as HTTP or SSL traffic. Thus, when load balancing application traffic, the BIG-IP system uses TMM switch interfaces. See also *management interface*.

TMM switch route

A Traffic Management Microkernel (TMM) switch route is a route that forwards traffic through the TMM switch interfaces and not the management interface.

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.

trusted CA file

A trusted CA file is a file containing a list of certificate authorities that an authenticating system can trust when processing client requests for authentication. A trusted CA file resides on the authenticating system and is used for authenticating SSL network traffic.

Type of Service (ToS) level

The Type of Service (ToS) level is another means, in addition to the Quality of Service (QoS) level, by which network equipment can identify and treat traffic differently based on an identifier.

Universal Inspection Engine (UIE)

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 configuration set (UCS)

A user configuration set is a backup file that you create for the BIG-IP system configuration data. When you create a UCS, the BIG-IP system assigns a **.ucs** extension to the filename. See also *archive.*

user role

A user role is a type and level of access that you assign to a BIG-IP system user account. By assigning user roles, you can control the extent to which BIG-IP system administrators can view or modify the BIG-IP system configuration.

virtual address

A virtual address is an IP address associated with one or more virtual servers managed by the LTM system.

virtual port

A virtual port is the port number or service name associated with one or more virtual servers managed by the LTM 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 an LTM system or other type of host server.

VLAN (virtual local area network)

A VLAN is a logical grouping of interfaces connected to network devices. You can use a VLAN to logically group devices that are on different network segments. Devices within a VLAN use layer 2 networking to communicate and define a broadcast domain.

VLAN group

A VLAN group is two or more VLANs that you put together into a VLAN group. A primary use of a VLAN group is to successfully route traffic when both the source and the destination hosts reside on the same network.

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 (virtual local area network).*

VLAN tag

An IEEE standard, a VLAN tag is an identification number inserted into the header of a frame that indicates the VLAN to which the destination device belongs. VLAN tags are used when a single interface forwards traffic for multiple VLANs.

WAP (Wireless Application Protocol)

WAP is an application environment and set of communication protocols for wireless devices designed to enable manufacturer-, vendor-, and technology-independent access to the Internet and advanced telephony services.

watchdog timer card

A watchdog timer card is a hardware device that monitors the BIG-IP system for hardware failure.

wildcard virtual server

A wildcard virtual server is a virtual server that uses an IP address of **0.0.0.0**, ***** or **"any"**. A wildcard virtual server accepts connection requests for destinations outside of the local network. Wildcard virtual servers are included only in Transparent Node Mode configurations.

WKS (well-known services)

Well-known services are protocols on ports 0 through 1023 that are widely used for certain types of data. Some examples of some well-known services (and their corresponding ports) are: HTTP (port 80), HTTPS (port 443), and FTP (port 20).

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X