coyote point | a subsidiary of FORTINET.

The recognized leader in proven and affordable load balancing and application delivery solutions

Application Delivery Controller

EQ/OS 10

Administration Guide

for Equalizer™ LX and GX Series

OS Version 10.3.2

April 15, 2015

Coyote Point Systems
A subsidiary of Fortinet, Inc.
56 Main Street
Millerton, NY 12546

Table of Contents

Chapter 1
Introduction

Subsections in this chapter include:

About Equalizer

The Equalizer Application Delivery Controller (ADC) is a high-performance switch that offers optimized availability, user experience, and performance of mobile, cloud-based and enterprise applications while increasing server efficiency and reducing cost and complexity in the data center. It features:

- Intelligent load balancing based on multiple, user-configurable criteria
- Non-stop availability with no single point of failure, through the use of redundant servers in a cluster and the optional addition of a failover (or backup) Equalizer
- Layer 7 content-sensitive routing
- Connection persistence using cookies or IP addresses
- Real-time server and cluster performance monitoring
- Server and cluster administration from a single interface
- SSL acceleration (on Equalizer models with Xcel SSL Hardware Acceleration)
- Data compression (on Equalizer models with Express Hardware GZIP Compression)
- Geographic load balancing

Typographical Conventions

The following typographical conventions appear throughout this guide:

- Text in “double quotes” indicates the introduction of a new term.
- Italic text is used primarily to indicate variables in command lines, and is also used to emphasize concepts while discussing Equalizer operation.
- **Boldface** text highlights GUI interface screen elements: labels, buttons, tabs, icons, etc., as well as data the user must type into a GUI element.
- `Courier` text denotes computer output: messages, commands, file names, directory names, keywords, and syntax exactly as displayed by the system.
- **`Bold courier`** text is text the user must type at the CLI prompt. Bold courier text in brackets -- indicates a keyboard key or key sequence that must be typed.
- Bold text sequences such as “**Cluster > Configuration > Settings**” are used to indicate the GUI controls a user needs to click to display the GUI form relevant to the task at hand. In the above example, the user would click on the Equalizer host name displayed at the top of the left navigational tree , click on the **Configuration** tab in the right pane, and then click on the **Settings** tab.

1. Numbered lists show steps that you must complete in the numbered order.

- Bulleted lists identify items that you can address in any order.

Note - A note box in the margin cites the source of information or provides a brief explanation that supports a specific statement but is not integral to the logical flow of the text.

The symbol on the left emphasizes a critical note or caution.

Attributions

Many of the icons used in the Web UI and reproduced in this manual are © Copyright 2013 FatCow Web Hosting. All rights reserved. These icons are licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/us/) and are used without modification.

Where to Go for More Help

These instructions are part of the product documentation delivered with Equalizer’s browser-based GUI. You can display the appropriate manual section for any interface screen by selecting **Help > Context** help from the menu at the top of the interface. The Help menu also contains links to the Release Notes for the currently running software version, and other documentation.

Hard copy documentation provided with every Equalizer includes the *Quick Start Guide* and the *Basic Configuration Guide*. These two documents are designed to help you get Equalizer out of the box and working with your first virtual clusters. The *Basic Configuration Guide* also contains a **Resource CD** with copies of all product documentation, including support documents that help you configure Equalizer for a variety of environments.

Register today to get access to the **Fortinet Support Portal:**

https://support.fortinet.com

Registration provides you with a login so you can access these benefits:

- **Support FAQs**: answers to our customer's most common questions.
- **Moderated Customer Support Forum**: ask questions and get answers from our support staff and other Equalizer users.
- **Software upgrades and security patches**: access to the latest software updates to keep your Equalizer current and secure.
- **Online device manuals, supplements, and release notes**: the latest Equalizer documentation and updates.
- **Links to additional resources**, and more.

Registration details can be found in "Registering Your Product" on page 64.

Chapter 2
Overview

Sections within this chapter include:

Intelligent Load Balancing

The Equalizer appliance functions as a gateway to one or more sets of servers organized into virtual clusters. When a client submits a request to a site that the appliance manages, it identifies the virtual cluster for which the request is intended, determines the server in the cluster that will be best able to handle the request, and forwards the request to that server for processing.

To route the request, the appliance modifies the header of the request packet with the appropriate server information and forwards the modified packet to the selected server. Depending on the cluster options chosen, it may also modify the headers in server responses on the way back to the client.

Equalizer supports clusters that route requests based on either Layer 4 (TCP or UDP) or Layer 7 (HTTP or HTTPS) protocols. Layer 4 is also referred to as the Transport Layer, while Layer 7 is referred to as the Application Layer. These terms come from the OSI and TCP/IP Reference Models, abstract models for network protocol design.

In general, Layer 4 clusters are intended for configurations where routing by the destination IP address of the request is sufficient and no examination of the request headers is required. Layer 7 clusters are intended for configurations where routing decisions need to be made based on the content of the request headers. the appliance evaluates and can modify the content of request headers as it routes packets to servers; in some cases, it can also modify headers in server responses on their way back to the client.

Basic Capabilities of Cluster Types Supported by Equalizer

Feature	**Cluster Type**			
	L4 UDP	**L4, L7 TCP**	**L7 HTTP**	**L7 HTTPS**
Load balancing policies	Round Robin, Static Weight, Adaptive, Fastest response, Least Connections, Server Agent, Custom			
Server failure detection (probes)	ICMP, TCP, Health Check	ICMP, TCP, ACV, Health Check		
Persistence	Based on IP		Using Cookies	
Server selection by request content (i.e., Match Rules)	No; load is balanced according to current load balancing policy.		Yes; load is balanced according to decisions made by examining request content.	
Load balanced protocols	Ideal for stateless UDP-based protocols, such as DNS and RADIUS; WAP gateways; NFS server clusters that provide a single-system image.	Ideal for stateful TCP-based protocols, such as HTTP, HTTPS, SMTP, FTP, LDAP/LDAPS and others.	HTTP	HTTPS
NAT and spoofing	Yes			

Regardless of cluster type, the appliance uses intelligent load balancing algorithms to determine the best server to receive a request. These algorithms take into account the configuration options set for the cluster and servers, real-time server status information, and information from the request itself. For Layer 7 clusters, user-defined match rules can also be used to determine the route a packet should take.

Real-Time Server Status Information

Equalizer gathers real-time information about a server’s status using ICMP Probes, TCP Probes, Active Content Verification (ACV), and Server Agents. ICMP and TCP Probes are the default probing methods.

ICMP Probes use Internet Control Message Protocol to send an "Echo request" to the server, and then wait for the server to respond with an ICMP "Echo reply" message (i.e., the Unix **ping** command). ICMP is a Layer 3 protocol. ICMP probes can be disabled via a global flag.

TCP Probes establish (and tear down) a TCP connection between the appliance and the server in a typical Layer 4 exchange of TCP SYN, ACK, and FIN packets. If the connection cannot be completed, the appliance considers the server down and stops routing requests to it. TCP probes cannot be disabled.

Active Content Verification (ACV) provides an optional method for checking the validity of a server’s response using Layer 7 network services that support a text-based request/response protocol, such as HTTP. When you enable ACV for a cluster, the appliance requests data from each server in the cluster (using an ACV Probe string)and verifies the returned data (against anACV Response string). If it receives no response or the response string is not in the response, the verification fails and it stops routing new requests to that server. See "Configuring ACV Health Checks" on page 567 for more information.

Note - ACV is not supported for Layer 4 UDP clusters.

Server Agent Probes enable the appliance to communicate with a user-written program (the agent) running on the server. A server agent is written to open a server port and, when the appliance connects to the port, the server agent responds with an indication of the current server load and performance. This enables adjustment of the dynamic weights of the server according to detailed performance measurements performed by the agent, based on any metrics available on the server. If the server is overloaded and you have enabled **server agent** load balancing, the appliance reduces the server’s dynamic weight so that the server receives fewer requests. The interface between the appliance and server agents is simple and well-defined. Agents can be written in any language supported on the server (e.g., perl, C, shell script, javascript, etc.). See "Configuring Server Agent Health Checks" on page 585for more information.

For those who have one or more VMware ESX Servers, VLB can be configured to use VMware’s status reporting to determine server status, and can also be configured to automatically manage VMware servers based on status information obtained from VMware.

Network Address Translation and Spoofing

The servers load balanced by Equalizer provide applications or services on specific IP addresses and ports, and are organized into virtual clusters, each with its own IP address. Clients send requests to the cluster IP addresses on the appliance instead of sending them to the IP addresses of the servers.

Central to the operation of any load balancer is the Network Address Translation (NAT) subsystem. On Equalizer, NAT is used as follows:

1. When Equalizer receives a client packet, it always translates the destination IP (the cluster IP) to the IP address of one of the server instances in a server pool. The server IP used is determined by the cluster's load balancing settings.
2. Depending on the setting of the cluster **spoof** option, Equalizer may also perform Source NAT, or SNAT.

   When the **spoof** option is enabled on a cluster, then SNAT is disabled: the NAT subsystem leaves the client IP address as the source IP address in the packet it forwards to the server. For this reason, the servers in a cluster with **spoof** enabled are usually configured to use Equalizer's IP address as their default gateway to ensure that all responses go through the appliance (otherwise, the server would attempt to respond directly to the client IP).

   When the **spoof** option is disabled on a cluster, then SNAT is enabled. Equalizer translates the source IP (the client IP) to one of the appliance's IP addresses before forwarding packets to a server. The servers will send responses back to the appliance's IP (so it is usually not necessary to set the appliance as the default gateway on the servers when **spoof** is disabled).

   Match rules can be used to selectively apply the **spoof** option to client requests. This is sometimes called selective SNAT. See "Creating a New Match Rule" on page 445.
3. When a server sends a response to a client request through Equalizer, the NAT subsystem always translates the source IP in the response packets (that is, the server IP) to the cluster IP to which the client originally sent the request. This is necessary since the client sent its original request to the cluster IP and will not recognize the server's IP address as a response to its request -- instead, it will drop the packet.

4. NAT can also be enabled for packets that originate on the servers behind Equalizer and are destined for subnets other than the subnet on which the servers reside -- on the appliance, this is called outbound NAT. This is usually required in dual network mode when reserved IP addresses (e.g., 10.x.x.x, 192.168.x.x) are being used on the internal interface, so that the recipients do not see reserved IP addresses in packets originating from the servers. When the global **outbound NAT** option is enabled, the appliance translates the source IP in packets from the servers that are not part of a client connection to the the appliance's Default VLAN IP address (the external interface IP address on the E250GX and legacy 'si' systems), or to the address specified in the server's **Outbound NAT** tab. Enabling **outbound NAT**, as a result, has a performance cost since the appliance is examining every outbound packet.

Note - When Equalizer is in single network mode, outbound NAT should be disabled. Since Equalizer resides on a single subnet, outbound NAT is not needed, and may cause unexpected behavior.

When Equalizer receives a packet that is not destined for a virtual cluster IP address, a failover IP address, a client IP address on an open connection, or one of its own IP addresses, the appliance passes the packet through to the destination network unaltered.

Load Balancing

Load balancing is based on the policy selected. The policies can be split up into two categories:

1. round robin
2. everything else

Round robin simply selects the next server in the list with no regard for how busy that server may be.

Other load balancing policies use proprietary algorithms to compute the load of a server and then select the server with the least load server.

Although the load balancing policies are proprietary, they use the following factors in their calculation:

- **Active connections** - The number of connections a server currently has active and the number of connections that it tends to have open.
- **Connection latency** - The amount of time that it takes a server to respond to a client request.
- **Health check performance values** - Depending on the health checks configured, this may be not used at all, or it can completely define how the load is calculated.

Once a load is calculated, Equalizer distributes incoming requests using the relative loads as weights.

sv00 Load = 50	sv01 Load = 50	sv02 Load = 50

Equalizer calculated loads, so the request distribution will be approximately equal

sv00 Load = 100	sv01 Load = 50	sv02 Load = 25

sv01 and sv02 above are uneven loads. sv01 is twice as loaded as sv02, so it will receive about half the requests.

The load calculations happen approximately every 10 seconds and server weights are adjusted accordingly. During that 10 second interval, the relative server loads remain the same, but probe and health check information is collected about the servers so that it can be used for the next calculation.

The load calculation works the same for Layer 4 and Layer 7 clusters (at the server-pool level – and these can be shared between all cluster types).

There are two additional variables for load balancing:

- **Hot spare** - if a server instance (in a server pool) is marked as a **Hot Spare**, it is not included in the pool of servers to select from unless every other non-hot-spare server is down. If a connection persists to this server, it will be placed back on this server.
- **Quiesce** - If a server instance (in a server pool) has been marked as **Quiesce**, it will not be included in the pool of servers to select from. Only previously existing (persistent) connections will be made to this server.

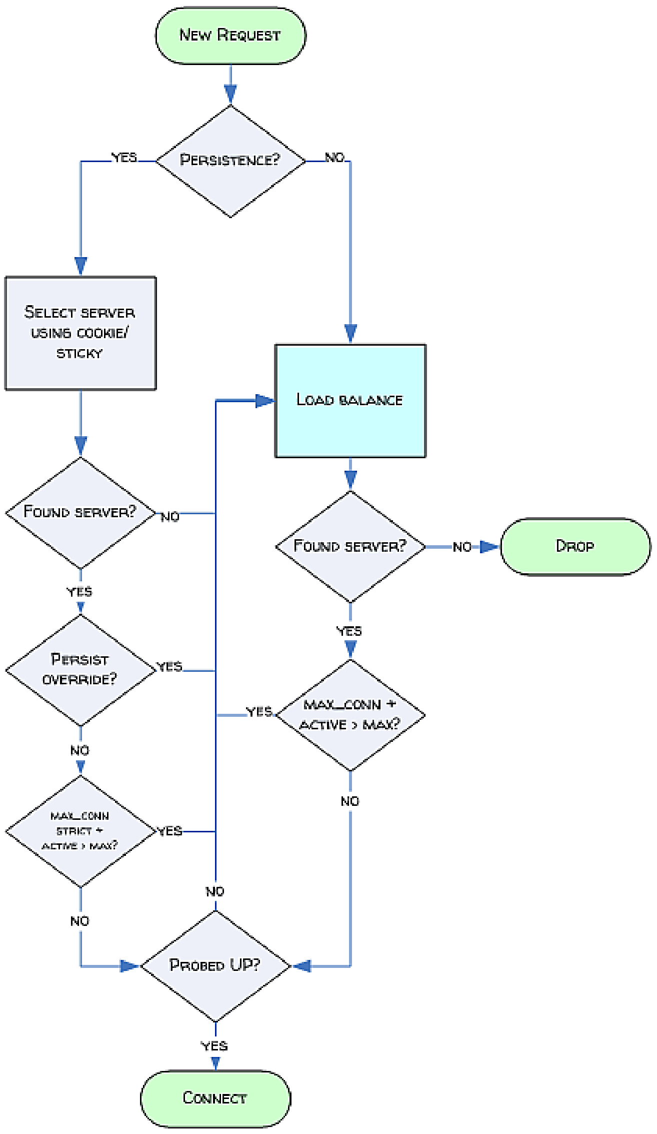
How a Server is Selected

The main functionality of Equalizer is to load-balance-- that is that when a request is received from a client an appropriate server for to connect the request with. The "appropriate" server is usually selected as part of a proprietary load balancing algorithm or via round-robin. Another factor in server selection is "persistence". If a client connection has persistence associated with it, the server to which the persists should be selected for load balancing If the server selected by persistence is not available, the appliance uses load balancing policy to select an alternate server.

Server Selection Process Flow

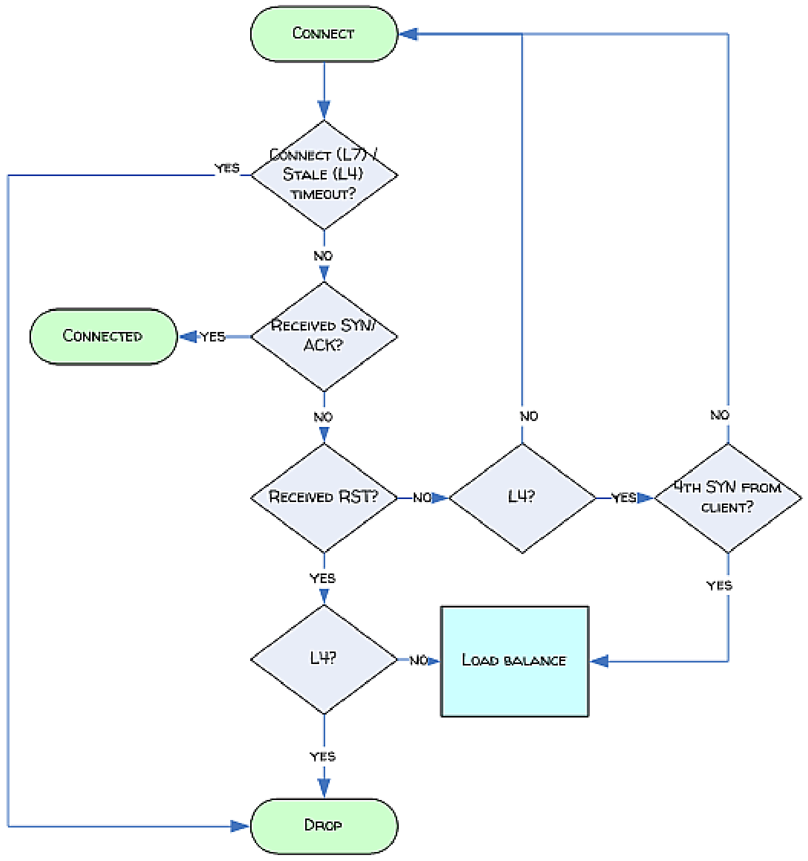
The figure below shows the server selection process. As describe above, this process depends on whether persistence is in use. Once a server is selected, Equalizer verifies that it isn't too busy (based on **max_connections**) and that it has been probed up. Then Equalizer tries to connect to it.

The figure below shows the connection establishment and server failover mechanism.



For Layer 7 clusters, the connection must be established within the **connect_timeout**. If we receive an active refusal (RST) from a server, we will repeat the load balancing process and choose another server. Otherwise we will continue trying to connect to the same server until the connect timeout expires.

Fore Layer 4 clusters, the connection must be established within the **stale_timeout**. Here, the appliance retries the same server 3 times, and then chooses another server on the 4th attempt. If the appliance receives an active refusal (RST) from a server, the connection is dropped.



Layer 7 Load Balancing and Server Selection

Equalizer’s support for Layer 7 content-sensitive load balancing enables administrators to define rules for routing HTTP, HTTPS, and special Layer 7 TCP requests, depending on the content of the request. Layer 7 load balancing routes requests based on information from the application layer. This provides access to the actual data payloads of the TCP/UDP packets exchanged between a client and server. For example, by examining the payloads, a program can base load-balancing decisions for HTTP requests on information in client request headers and methods, server response headers, and page data.

Equalizer’s Layer 7 load balancing allows administrators to define rules in the administration interface for routing HTTP and HTTPS requests according to the request content. These rules are called match rules. A match rule might, for example, route requests based on whether the request is for a text file or a graphics file. For example, you may want to:

- load balance all requests for text files (html, etc.) across servers A and B
- load balance all requests for graphics files across servers C, D, and E
- load balance all other requests across all of the servers

Match Rules are constructed using match functions that make decisions based on the following:

- HTTP protocol version; for HTTPS connections, the SSL protocol level the client uses to connect.
- Client IP address
- Request method (GET, POST, etc.)
- All elements of the request URI (host name, path, file name, query, etc.)
- Pattern matches against request headers

Match functions can be combined using logical constructs (AND, OR, NOT, etc.) to create extremely flexible cluster configurations. See "Managing Match Rules" on page 443 for an overview of Match Rules, a complete list of match functions, and usage examples.

Persistence

Persistence refers to the ability of a load-balancer (or other traffic management solution) to maintain a virtual connection between a client and a specific server.It is often referred to in the application delivery networking world as "stickiness" .The persistence of session data is important when a client and server need to refer to data previously generated again and again as they interact over more than one transaction, possibly more than one connection. Whenever a client places an item in a shopping cart, for example, session data (the item in the cart, customer information, etc.) is created that potentially needs to persist across many individual TCP connections before the data is no longer needed and the session is complete.

It’s important to note that session persistence is managed by the server application, not Equalizer. the appliance provides server persistence so that a persistent connection between a particular client and a particular server can be maintained; this supports a client-server session where session data is being maintained on the server for the life of the connection. In other words, whether you need to enable persistence on the appliance depends on the application you are load balancing.

Equalizers have no knowledge of the fact that the user has placed something in a shopping cart, logged into a web application, requested a file from shared storage, or made a "post" in a front end presentation server that has been written to a database. Basically, a "state" has been created in the load balanced application of which Equalizer is not aware. What the appliance does know is that a specific client has been load balanced to a specific server in one of its virtual clusters. With this knowledge, it can track that information and send that client back to the same server they were connected the first time.

Layer 7 Persistence

Equalizer provides server or connection persistence using cookies in Layer 7 HTTP and HTTPS clusters. The following paragraphs explain connection persistence provided by the appliance, and its relationship to session persistence.

When a request from a client that has not previously connected to the cluster is received by Equalizer, it is load balanced according to the current server load values as described in "Load Balancing" on page 28.

However, when a client has existing persistence to a server, Equalizer attempts to put the client back on that server.

Equalizer can use cookies or a server’s IP address to maintain a persistent session between a client and a particular server. A cookie, used in Equalizer HTTP and HTTPS clusters contains the identity of the server that should be used. When a client connects to the cluster for the first time, Equalizer injects this cookie into the response data. The client’s browser is then responsible for presenting this cookie back to Equalizer. If Equalizer finds this cookie in the client’s request, it connects to the server listed.

EQ/OS 10 features "fallback persistence" where Equalizer provide a secondary persistence option where if, for example, a cookie response is not received, a secondary, or "fallback" option can be used. As an example, if two persist methods are listed (e.g., Cookie 1:Cluster IP, Server IP /Port and Source IP)- if a cookie is found- the cookie will be used, otherwise the Source IP of the incoming connection will be used. If the server with which a client has an unavailable persistent session, Equalizer automatically selects a different server. Then, the client must establish a new session; Equalizer stuffs a new cookie in the next response. Details and scenarios are presented in "Fallback Persistence Scenarios" on page 393.

Layer 4 Persistence

For Layer 4 TCP and UDP clusters, Equalizer support IP address-based persistent connections. With a sticky connection option enabled, Equalizer identifies clients by their IP addresses when they connect to a cluster. It then routes requests received from a particular client during a specified period of time to the same server pool in the cluster.

A sticky timer measures the amount of time that has passed since there was a connection from a particular IP address to a specific cluster. The sticky time period begins to expire as soon as there are no longer any active connections between the client and the selected cluster. Equalizer resets the timer whenever a new connection occurs. If the client does not establish any new connections to the same cluster, the timer continues to run until the sticky time period expires. At expiration, Equalizer handles any new connection from that client like any other incoming connection and routes it to an available server based on the current load balancing policy.

To correctly handle sticky connections from ISPs that use multiple proxy servers to direct user connections, Equalizer supports sticky network aggregation, which uses only the network portion of a client's IP address to maintain a persistent connection. Sticky network aggregation directs the user to the same server no matter which proxy he or she connects through.

Equalizer can also be configured to ensure that it directs requests from a particular client to the same server pool even if the incoming connection is to a different cluster. When you enable inter cluster stickiness for a Layer 4 cluster, Equalizer checks the cluster for a sticky record as it receives each connection request, just like it does for ordinary sticky connections. If the appliance does not find a sticky record, it proceeds to check all of the other clusters that have the same IP address. If it still does not find a sticky record, it connects the user based on the current load balancing policy.

Why a Server May Not Be Selected

There are several reasons that a server may not be selected by Equalizer:

1. The various configured health checks within Equalizer have detected that a server is "down". If a server is marked "down" by a health check, it is immediately removed from the pool of servers available for load balancing.
2. For Layer 4 clusters, if health checks have not yet detected that a server is "down", and Equalizer is unable to establish a cluster connection with the server, it will keep retrying the same server until the 4th SYN packet received from the client and then use the load balancing policy to select a new server. The whole connection establishment must complete within the configured **stale_timeout** time frame or the connection will be dropped. If Equalizer chooses a different server than the persistence record, it overwrites the persistence record to use the new server for next time.

Note - Most clients will not have time to retry 3 times (send a 4th SYN packet) within the default 10 second stale timeout window. Therefore the connection will be dropped and the process will be started over again when the next SYN is received. (The 1st SYN would be at time 0, the 2nd at time 3, the 3rd at time 9... so the 4th would not happen before 10 seconds).

3. For Layer 7 clusters, if health checks have not yet detected that the server is "down" but Equalizer is unable to establish a cluster connection with the server, it will wait the configured **connect_timeout** time frame and then drop the connection so that the client can retry. If it receives an active refusal from the server (RST packet), Equalizer will choose a different server and overwrite the persistence record to use the new server for next time.
4. Maximum connections - If the **Maximum Connection**s option is used for a server instance, and the server already has that many active connections, it will not be used. This means that it will not be included in the list of servers to select for load balancing. If persistence is in use, the **strict_max_connections** flag specifies whether to persist to a server which already has more active connections than **max_connections** or to load balance to a new server.
5. If the **persist_override** flag is selected in a server instance, and that server is selected by the load balancing policy, the client will not persist to this server even if persistence is enabled at the cluster level.

Chapter 3
What's New

Subsections in this chapter include:

What's New

The list below contains changes in this documentation since the previous release. For additional upgrade information, refer to the Release Notes available with the firmware.

For 10.3.2d:

1. **Documentation Enhancements:**

   Initial Setup and Basic Configuration section has been replaced by two new sections:

   - **Basic Configuration** - this section provides instructions for basic configuration that includes:
     - setting up a password
     - creating a VLAN
     - adding a subnet
     - configuring services on the subnet
     - setting up routing
     - configuring interfaces
     - replacing default certificates, keys, and cipherspecs.
   - **Enabling Network Services** - this section provides instructions for setting up:
     - global services
     - enabling VLAN subnet network services

2. **New Feature Descriptions:**

   HTTP Caching -- The HTTP Caching feature caches recently-accessed files in Equalizer memory for easy retrieval by clients. While cached, requests for these files will be served by the Equalizer instead of making queries to the servers behind it. The result is improved response times for your clients and reduced loads on your servers so that they can reserve more resources for delivering content quickly.

   FortiDirector GSLB Access - The Equalizer GUI now features convenient access to FortiDirector™ HTTP and DNS Redirect Statistics. The feature allows you to monitor daily statistics in numerical format and in a graphical plot.

3. **Feature Update Descriptions:**

   Additional Header Editing Functions have been added to the "HTTP and HTTPS Header Editing" appendix. The new functions are:

   str_find_item()
   str_delete(locator)
   str_find_item_by_name()
   str_insert_item()
   str_find_item_value()
   str_delete_item()
   str_insert()
   str_replace_item()
   str_replace()
   str_replace_item_value()

   Default UDP Health Check- added to "Health Checks". This health check is attached to a UDP server pool when the first UDP server instance is added to a server pool.

   Probe Coalescing - added to "Health Checks". This describes how the probes on two attached health checks of the same type, with the same parameters,coalesce into one .

4. **Command, Configuration, and Miscellaneous Change Descriptions:**

   Server Pool CLI Command summary changes -- the CLI "Server Pool Command Summary" has been modified to include the new caching commands.

   Diagnostic CLI Command summary changes -- added additional caching diagnostic commands to:

   - Flush a single entry from the server pool cache using an ID (obtained by displaying all cache entries)
   - Flush all entries from a server pool cache.
   - Flush all entries from all server pool caches.
   - Delete all the cache entries of a particular server pool.
   - Display all the cache entries of a particular server pool.

Chapter 4
Installation

Sections within this chapter include:

Hardware Installation

To install Equalizer, proceed with the following:

1. Carefully remove the rack-mount enclosure and cables from the shipping container.
   Save the original packaging in case you need to ship the appliance for any reason, such as sending it in for warranty service. The chassis does not contain any parts that you can service. If you open the chassis or attempt to make repairs, you may void your warranty.

2. Place the appliance in its intended position in an EIA equipment rack or on a flat surface.
3. Connect a serial terminal or a workstation running terminal emulator software to the serial port on the front panel of appliance. The serial cable is supplied with the unit.
4. Connect the appliance to the network with a quality category 5 (Cat 5E) network cable:

To use Equalizer as an intermediary between an external and internal network, connect it to the external network using one of the RJ-45 ports labeled 1 or 2 on the front panel. Connect the appliance to the internal network using one or more of the ports numbered 3 and above.

For a single-network (one subnet) topology, connect Equalizer to the network and the servers using one of the numbered RJ-45 ports numbered 3 and above on the front panel.

5. Connect the appliance to an appropriate power source using the supplied power cord, which plugs into the 3-pin connector on the rear of the enclosure. This system uses an auto-sensing power supply that can operate at 50Hz or 60Hz, 110-240 VAC input.
6. Turn on the power using the switch on the rear panel. After the appliance boots up the following link lights should be visible. Refer to "Interfaces" on page 296 or Interface Commands for additional information on the interface lights and settings.

UL/cUL & CE/CB Safety Warnings and Precautions

Risk of explosion if battery is replaced by an incorrect type. Dispose of used batteries according to your local regulations.

Switzerland: Annex 4.10 of SR814.013 applies to batteries.

Statement in Chinese text:

警告

本電池如果更換不正確會有爆炸的危險

請依製造商說明書處理用過之電池

Rack mount instructions:

Elevated Operating Ambient - If installed in a closed or multi-unit rack assembly, the operating ambient temperature of the rack environment may be greater than room ambient. Therefore, consideration should be given to installing the equipment in an environment compatible with the maximum ambient temperature (Tma) specified by the manufacturer.

Reduced Air Flow - Installation of the equipment in a rack should be such that the amount of air flow required for safe operation of the equipment is not compromised.

Mechanical Loading - Mounting of the equipment in the rack should be such that a hazardous condition is not achieved due to uneven mechanical loading.

Circuit Overloading - Consideration should be given to the connection of the equipment to the supply circuit and the effect that overloading of the circuits might have on overcurrent protection and supply wiring. Appropriate consideration of equipment nameplate ratings should be used when addressing this concern.

Reliable Earthing - Reliable earthing of rack-mounted equipment should be maintained. Particular attention should be given to supply connections other than direct connections to the branch circuit (e.g. use of power strips).

Grounding:

Ensure your product is connected and properly grounded to a lightning and surge protector.

WAN or LAN connections that enter the premises from outside the building should be connected to an Ethernet CAT5 (10/100 Mb/s) surge protector.

Shielded Twisted Pair (STP) Ethernet cables should be used whenever possible rather than Unshielded Twisted Pair (UTP).

Do not connect or disconnect cables during lightning activity to avoid damage to your product or personal injury.

Electrostatic discharge (ESD) can damage equipment. Only perform the procedures described in this document from an ESD workstation. If no such station is available, you can provide some ESD protection by wearing an anti-static wrist strap and attaching it to an available ESD connector or other bare metal object.

Regulatory Notices

For Class A – Regular ITE products

Federal Communication Commission (FCC) – USA

This device complies with Part 15 of FCC Rules. Operation is subject to the following two conditions:

This device may not cause harmful interference, and

This device must accept any interference received, including interference that may cause undesired operation.

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

Warning: Any changes or modifications to this product not expressly approved by the party responsible for compliance could void the user’s authority to operate the equipment.

Industry Canada Equipment Standard for Digital Equipment (ICES) – Canada

CAN ICES-3 (A) / NMB-3 (A)

This digital apparatus does not exceed the Class A limits for radio noise emissions from digital apparatus set out in the Radio Interference Regulations of the Canadian Department of Communications.

Le présent appareil numérique n’emet pas de bruits radioélectriques dépassant les limites applicables aux appareils numeriques de la classe A préscrites dans le Règlement sur le brouillage radioélectrique édicte par le ministère des Communications du Canada.

Voluntary Control Council for Interference (VCCI) – Japan

この装置は、クラスA情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。VCCI－A

Bureau of Standards Metrology and Inspection (BSMI) – Taiwan

這是甲類的資訊產品，在居住的環境中使用時，可能會造成射頻干擾，在這種情況下，使用者會被要求採取某些適當的對策。

China

此为A级产品，在生活环境中，该产品可能会造成无线电干扰。在这种情况下，可能需要用户对其采取切实可行的措施。

European Conformity (CE) – EU

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

IMPORTANT: Switzerland: Annex 4.10 of SR814.013 applies to batteries.

Power Requirements

The unit’s power supply is rated at 100-240 VAC auto selecting 60/50 Hz @ 4.0A.

Operating Environment

- **Temperature**: 40 - 105 °F, 5 - 40 °C. (GX Series)| 32 - 104°F, 0 - 40°C (LX Series)
- **Humidity**: 5 - 90%, non-condensing.

Regulatory Certification

Please see the product data sheets on the Coyote Point Website (www.coyotepoint.com) for product certification details.

Setting Up a Terminal or Terminal Emulator

When you set up Equalizer for the first time, you must use a serial connection in order to configure the appliance's network with the **eqcli** interface. Connect the serial port on the to the serial port on a terminal, or any system (such as a Windows or Unix PC) running terminal emulation software.

Configure your terminal or terminal emulator software to use the following settings:

- 9600 baud (GX Series) 38400 (LX Series)
- 8 data bits
- no parity
- one stop bit
- VT100 terminal emulation
- ignore hang-ups (if supported); this allows a single terminal session to continue running even if the appliance restarts

On Windows systems, you can use the Windows built-in terminal emulator, **HyperTerminal**, or the **Tera Term Pro** terminal emulator to log in over the serial port. On Unix systems, you can use the **cu** (1) command or any other Unix serial communication program.

If you use **HyperTerminal**, in addition to the settings shown above, select **File > Properties > Settings** from HyperTerminal's menu, select **VT100** in the **Emulation** drop-down box, and then **Terminal Setup** to enable these options:

- keyboard application mode
- cursor keypad mode

Tera Term is freely available at:

```
http://hp.vector.co.jp/authors/VA002416/teraterm.html
```

Chapter 5 Configuring Access

Sections within this chapter include:

Default Login

Equalizer's default login credentials for both the CLI and GUI are:

Username:`touch`

Password:`touch`

For security, you should change the login for the `touch` user the first time you log in. You can do this by logging into the CLI, entering the following command, and following the command prompts:

```
eqcli > user touch password
```

Creating Additional Logins

You can create additional administrative logins and assign specific permissions to individual logins, if desired.

Serial Access

Serial access is provided via the serial port on Equalizer’s front panel. A serial connection is required for activities during which the appliance may lose network connectivity. This includes:

- Configuring network connectivity for the first time
- Performing upgrades of the EQ/OS software and switch firmware
- Re-configuring network access for services such as HTTP and SSH, when you cannot login over the network interfaces currently configured or you are changing the network interfaces that will provide those services.

Chapter 6
Basic Configuration

Subsections include:

Basic Configuration

The following procedure is an example of how to configure VLANs using the Equalizer CLI. You must configure VLANs using the CLI so that you can ultimately use the GUI. Follow the steps below to get Equalizer onto your network .

1. Log in using the default administrative user name, **touch**:

```
Username: touch
Password: touch
Login successful.

       EQ/OS 10.3.2d
       Copyright 2015 Fortinet, Inc.

Welcome to Equalizer!
eqcli >
```

2. Change the password for the "touch" login. Enter:

```
eqcli > user touch passwd
```

   Follow the command prompts to create a new password.

3. Create a VLAN , enter a command like the following:

```
eqcli > vlan vlname vid vlan_ID
```

   Replace ***vlname*** with the VLAN name and **vlan_ID** with the VLAN ID number (1-4094). If you are using untagged VLANs (common in many sites), the VLAN ID can be any number not used on another VLAN. If you are using tagged VLANs, check with your network Administrator for the correct **vid** to specify.

4. Add a subnet to the VLAN you just created. You'll need to specify the subnet IP address, which is the load balancer's address on this network. It must be an IPv4 or IPv6 address in CIDR format (e.g., 172.16.0.200/21).

   Enter the following command syntax:

   ```
   eqcli > vlan vlname subnet subnetname ip cidr_ip
   ```

   In the command above, ***vlname*** is the VLAN name, ***netname*** is the name of the subnet, ***cidr_ip*** is the CIDR format IP address. For example:

   ```
   eqcli > vlan 172net subnet sn01 ip 172.16.0.200/21
   ```

5. Configure services on thesubnet. For our example, we'll enable SSH login and the GUI over HTTP on the **172net** VLAN.

   ```
   eqcli > vlan 172net subnet sn01 services http,https,ssh
   ```

6. Configure routing on the VLAN including a default route and gateway routes. In the example below, **0/0** is the default route and **172.16.0.1** is the gateway, which is an unadorned IP addresses. In this scenario, all packets fordestinations are to be sent via this routes.

   ```
   eqcli > vlan 172net subnet sn01 route 0/0 gw 172.16.0.1
   ```

   Refer to the webhelp if you need more help setting up your initial VLAN and subnet: go to www.coyotepoint.com, move your mouse over the **Support** link near the top of the screen, and choose **Manuals** from the drop down list.

5. Associate an interface instance with the VLAN. Here we assume that you are using the port labeled "1" on the front panel. Enter one of the following commands, depending on whether the VLAN you created above is untagged or tagged (ask your network administrator if you are unsure).

   eqcli > **vlan *vlname* ifi if01 type untagged** **or** eqcli > **vlan *vlname* ifi if01 type tagged**

6. Connect the port or ports you configured on the VLAN to the network using a standard Ethernet cable with RJ-45 connectors. To confirm that the interface has come up, use the following command:

```
eqcli > show interface
Interface  Duplex Mode  Speed  Status
ge01       NA           NA     Link Down
ge02       NA           NA     Link Down
ge03       NA           NA     Link Down
ge04       NA           NA     Link Down
ge05       NA           NA     Link Down
ge06       NA           NA     Link Down
ge07       NA           NA     Link Down
ge08       NA           NA     Link Down
xe01       full         10G    Link Up
xe02       full         10G    Link Up
mgmt       full         100M   Link Up
eqcli >
```

   The above example shows the appropriate output assuming that you are using the port labeled “1” on the front panel.

   You should now be able to use the “ping” command from a workstation on the same subnet to reach the subnet IP address configured above.

7. Connect the appliance to your network using the VLAN ports that you set up above. You should now be able to display the GUI by pointing a browser at this URL:

`http://`*`VLAN_IP_addr`*

Substitute the VLAN IP address, as in this example using the IP address from Step 3.

Replacing the Default Certificate, Key, and Cipherspec

Using Equalizer's Remote Management commands in the CLI, you can replace the default certificate, key, and cipher spec that are used with HTTPS services on subnets with custom certificates, keys and cipher specs.

The process includes:

- Uploading the custom certificate and key file to the file store.
- Entering the certificate (and key file) to be used with HTTPS services.
- Replacing the default cipherspec with the a custom cipher spec.
- Setting the encryption level to use in the communications between the client and the ADC.

Uploading the Custom Certificate and Key File

Enter the following to upload a certificate and key file:

1. Enter the name of the new certificate and upload it as follows:

```
eqcli > certificate certificatename certfile URL
```

   where ***URL*** downloads the **`certfile`** using **ftp://** or **http://** protocol.

2. Upload the new key file. The key file must have the same name as the certificate.

```
eqcli > certificate certificatename keyfile URL
```

   where ***URL*** downloads the **`keyfile`** using **ftp://** or **http://** protocol.

Entering the Certificate and Key file to be Used with HTTPS Services

3. Set the **`certfile`** and **`keyfile`** to use using the CLI remote management commands. The **`keyfile`** has the same name as the **`certfile`** and will be used automatically.

```
eqcli remote-mgmt certificate certificatename
```

4. Now view the remote management configuration. The example that follows shows that the `custom certificate` has been added:

```
eqcli > show remote-mgmt
Options           Value
Cipherspec        AES128-SHA:DES-CBC3-SHA:RC4-SHA:RC4-MD5:AES256-SHA:!SSLv2
Certificate       custom certificate
Protocols         tls10

eqcli >
```

Replacing the Default Cipherspec with a Custom Cipherspec

5. Enter the custom cipherspec as follows:

```
eqcli > remote-mgmt cipherspec cipherspec
```

where ***`cipherspec`*** is the new, custom `cipherspec` to be used.

Setting the Encryption Levels

6. Configure the encryption levels that will be used in communications between the client and the ADC. The default encryption level is TLSv1.0 (`tls10`).

```
eqcli > protocol protocol
```

where ***`protocol`*** can be **`sslv3`**, **`tls10(default)`**, **`tls11`**, or **`tls12`**. The protocols in the syntax can be delimited by "," or "|".

You can also turn off one of the protocols in the list by prefixing with "!". For example if you have configured all of the encryption levels to be used and want to remove `tls12`, enter `eqcli >` **`protocol !tls12`**. **`tls12`** would then be removed from the list. The client and ADP will use the highest level available when multiple formats are specified.

Reapplying the Default Certificate, Cipherspec and Protocols

To reapply the defaults for Cipherspec, Certificate or Protocol, enter any of the following:

```
eqcli > no remote-mgmt {cipherspec|certificate|protocol}
```

Chapter 7
Enabling Network Services

Subsections include:

Enabling Global Network Services

In order to access Equalizer over the network, network services must be enabled globally (that is, for all subnets) and on the specific subnets over which you want to provide access.

The Global Services settings provide a convenient way to enable and disable services on all subnets, should the need arise. For example, when you are upgrading or performing a system backup, it may be desirable to use the serial connection and disable all network services to ensure that no other administrative users are accessing the system.

By default, all services are enabled globally:

Global Services Using the CLI

In the CLI, global services settings are managed using the global **services** parameter (see "Global Commands" on page 164).

Global Services Using the GUI

Follow these steps to set the system **Hostname**, **Date & Time**, and **DNS** in the GUI:

1. Log into the GUI to perform additional configuration. On a workstation that is on the same subnet that you configured above, or on a network that can route to and from the subnet you configured above, open a web browser and enter Equalizer’s IP address into the browser’s address bar. At the GUI login screen, enter the `touch` user name and the password that you assigned earlier, and click **Login**. The Welcome screen for the Equalizer GUI appears on the right pane.
2. On the left navigational pane, select **System > Global > Parameters**configuration tab in the left pane. The Global **Parameters** screen will be displayed on the right pane.
   a. Set the system **Hostname** to a name that is unique on your network.
   b. Optionally set up to three Domain Name Servers.
   c. Click on **Commit**.to save the settings.
3. Click on the arrow (▶) beside **Maintenance** to expand the branch and select **Date & Time** to display the **Data & Time** display on the right pane. In the **Set Timezone** field, locate your time zone in the drop-down box and click **Commit**.
4. Do one of the following:
   a. If Equalizercan connect to the Internet and you defined at least one DNS server above, you can configure the Network Time Protocol (NTP). In the **Automatically Set Date and Time** field, type the name of an NTP Server into the text box and turn on the **Enable NTP Synchronization** check box. Click **Commit**.
   b. Otherwise, set the date and time manually by modifying the contents of the **Date** field. Click **Commit**.

The following global services settings are supported:

GUI **(CLI)**	**Global Service**
HTTP **(`http`)**	HTTP GUI service; when enabled, the Equalizer GUI will listen on all subnets on which HTTP services are enabled.
HTTPS **(`https`)**	HTTPS GUI service; when enabled, the Equalizer GUI will listen on all subnets on which HTTPS services are enabled.
SSH **(`ssh`)**	SSH login service; when enabled, SSH login will be permitted on all subnets on which SSH services are enabled.
SNMP **(`snmp`)**	SNMP (Simple Network Management Protocol) service; when enabled, SNMP will accept connections on all subnets on which SNMP services are enabled.

Enabling VLAN Subnet Network Services

By default, no network services are enabled when a VLAN subnet is created. They must be specifically enabled before you can access Equalizer over a subnet:

VLAN Subnet Network Services using the CLI

In the CLI, subnet network services are enabled using the **services** parameter in the **subnet** context. See "VLAN and Subnet Commands" on page 231.

VLAN Subnet Network Services using the GUI

In the GUI, click on the **System** configuration tab on the left pane.

- Clicking on the arrow (▶) next to **Network** expands the branch and provides access to Interfaces, VLANs, and Tunnels displays on the right pane.
- Clicking on the arrow (▶) next to **VLANs** expands the branch to display all configured VLANs.
- Clicking on the arrow (▶) for each VLAN expands the branch to display the configured subnets. Click on each subnet to display the subnet **Configuration** screen on the right pane for the selected subnet. Failover subnet services are configured for the selected subnet as well by clicking on the **Failover** tab on the right pane.

The following subnet network services settings are supported:

CLI	GUI	Network Service
http	**HTTP**	HTTP GUI service; when enabled, the Equalizer will listen for HTTP connections on Equalizer’s IP address on the subnet. The global HTTP GUI service must also be enabled.
https	**HTTPS**	HTTPS GUI service; when enabled, the Equalizer will listen for HTTPS connections on Equalizer’s IP address on the subnet. The global HTTPS GUI service must also be enabled.
ssh	**SSH**	SSH log in service; when enabled, SSH log in will be permitted on Equalizer’s IP address on the subnet. The global SSH service must also be enabled.
snmp	**SNMP**	SNMP (Simple Network Management Protocol) service; when enabled, SNMP will accept connections on Equalizer’s IP address on the subnet. The global SNMP service must also be enabled.
envoy	**Envoy**	Envoy DNS service; when enabled, Envoy will accept DNS lookup connections on Equalizer’s IP address on the subnet. The global Envoy service must also be enabled.
envoy_agent	**Envoy Agent**	Envoy Agent health check service; when enabled, Envoy health checks will be performed on the subnet using Equalizer’s IP address on the subnet as the source IP. The global Envoy Agent service must also be enabled.

CLI	GUI	Network Service
fo_http	**Failover HTTP**	Failover HTTP GUI service; when enabled, the Equalizer will listen for HTTP connections on Equalizer’s Failover IP address (if configured) on the subnet. The global HTTP GUI service must also be enabled.
Click on the Failover tab to enable or disable the following services:		
fo_https	**Failover HTTPS**	Failover HTTPS GUI service; when enabled, the Equalizer will listen for HTTPS connections on Equalizer’s Failover IP address (if configured) on the subnet. The global HTTPS GUI service must also be enabled.
fo_ssh	**Failover SSH**	Failover SSH log in service; when enabled, SSH log in will be permitted on Equalizer’s Failover IP address (if configured) on the subnet. The global SSH service must also be enabled.
fo_snmp	**Failover SNMP**	Failover SNMP service; when enabled, SNMP will accept connections on Equalizer’s Failover IP address (if configured) on the subnet. The global SNMP service must also be enabled.
fo_envoy	**Failover Envoy**	Failover Envoy DNS service; when enabled, Envoy will accept DNS lookup connections on Equalizer’s Failover IP address (if configured) on the subnet. The global Envoy service must also be enabled.
fo_envoy_agent	**Failover Envoy Agent**	Failover Envoy Agent health check service; when enabled, Envoy health checks will be performed on the subnet using Equalizer’s Failover IP address on the subnet as the source IP. The global Envoy Agent service must also be enabled.

Chapter 8
Registering Your Product

Sections within this chapter include:

Registering Your Product

Fortinet customer services (such as firmware updates and technical support) require product registration. Take a moment now to register your product at the Fortinet Customer Service and Support web site:

> https://support.fortinet.com

Before you can register, you will need:

1. **Access to a new or existing Support Account**. Information on how to create and manage a support account is provided in the Fortinet Support Portal User Guide. If your organization already has an account, obtain the user name and password information from your local account administrator to log in.
2. **The serial number of the unit you want to register.** You can find this information using either the CLI or the GUI after powering up your appliance:
   a. To use the CLI, log in to the CLI (over the serial console or, if networking is configured, using SSH over an appropriately configured subnet) and enter the following CLI command:

      ```
      eqcli > version.
      ```

      Record the **`System Serial Number`** from the command output.
   b. If networking is configured and the GUI has been enabled on a subnet., you can also get the serial number from the **System Information** widget on the GUI dashboard. The Dashboard appears automatically when you log into the GUI.

Once you have obtained both the login credentials of a support account and the System Serial Number of the unit to register, do the following:

1. Log in to https://support.fortinet.com using the login credentials obtained above.
2. Follow the instructions provided in the Registration Frequently Asked Questions under the heading "How do I register a Fortinet device?" to register your Equalizer. When requested, enter the **System Serial Number** you obtained above into the appropriate form. Once registration is completed, the appliance serial number and other information will appear in the FortiCare Registration area.

3. Your Equalizer system is now registered. If your system can connect to the internet, you can now update the support information displayed in the CLI and GUI by doing one of the following:

   a. In the CLI, enter the following to update the support information on your unit:

      eqcli > **forticare registration**

      View the updated **`Support information`** (including **`Last refresh date`**, **`Support end`**, and **`Email`**) by entering:

      eqcli > **`version`**

   b. In the GUI, select the **System** configuration tab on the left navigational pane and then click on **Global > Dashboard**. The **System Information** widget on the right pane will indicate the Support information (including **Last refresh date**, **Support end**, and **Email**). Click on the **Refresh** button to update the registration information.

Note - FortiCare information is not provided with E250GX systems in either the CLI or GUI.

Note - The registration information does not update automatically in either the CLI or the GUI; you must use either the CLI "**`forticare registration`**" command or the **Refresh** button in the Dashboard's **System Information** widget to update.

Chapter 9

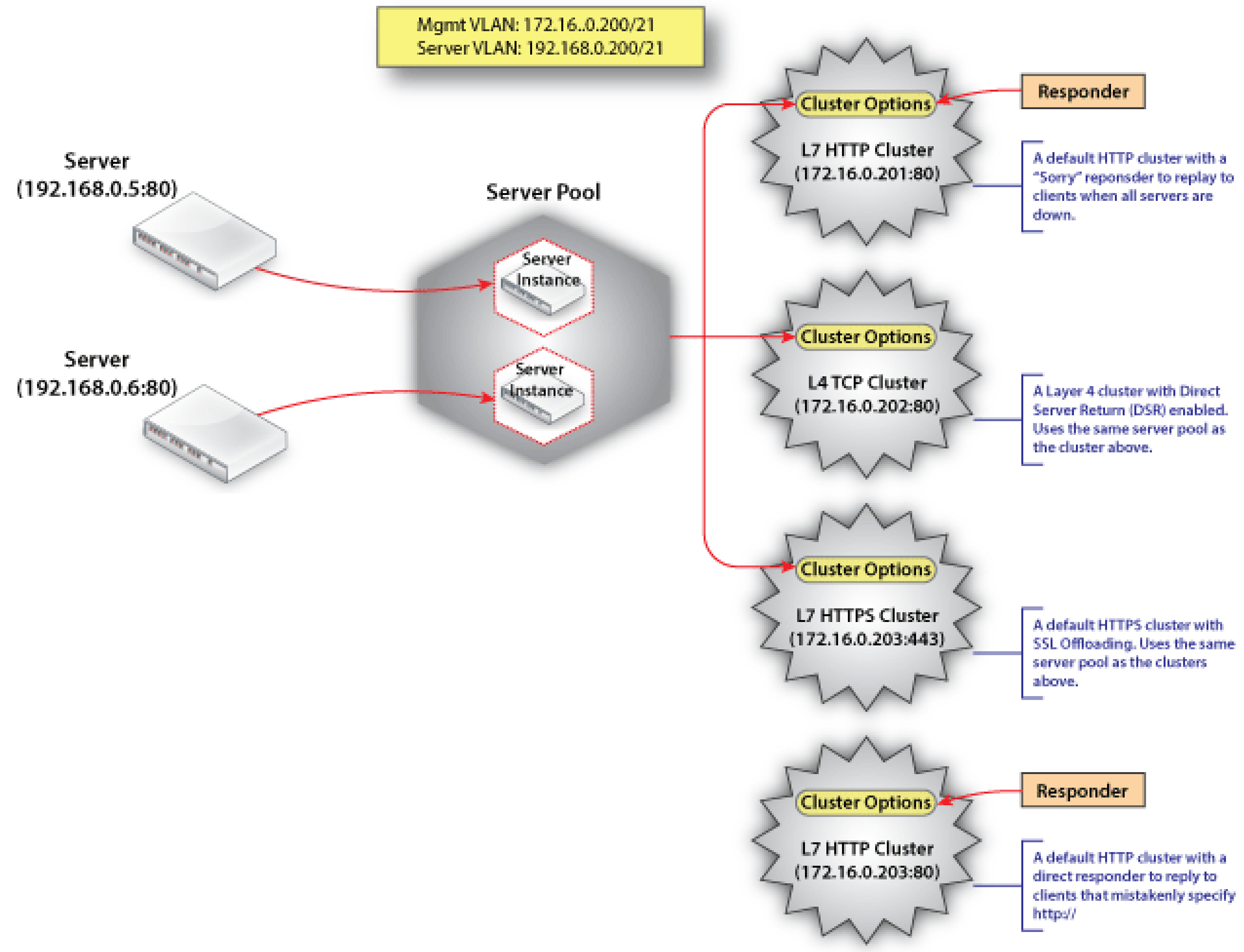
Sample Configuration

Sections within this chapter include:

Sample Configuration

The following is an example of basic Equalizer configuration. It includes the configuration VLANs and other load balancing objects such as servers, server pools, clusters and responders. The example assumes that Equalizer is in a “factory installed” state: no customer configuration has been performed on the unit. The sample configuration we’ll create is pictured in the illustration below.

Note - The IP addresses shown in the sample configuration below are for demonstration purposes only! Consult with your network administrator for the proper IP addresses to use in your configuration.


Mgmt VLAN: 172.16..0.200/21
Server VLAN: 192.168.0.200/21
Server (192.168.0.5:80)
Server (192.168.0.6:80)
Server Pool
Server Instance
Server Instance
Cluster Options
L7 HTTP Cluster (172.16.0.201:80)
Responder
A default HTTP cluster with a “Sorry” reponsder to replay to clients when all servers are down.
Cluster Options
L4 TCP Cluster (172.16.0.202:80)
A Layer 4 cluster with Direct Server Return (DSR) enabled. Uses the same server pool as the cluster above.
Cluster Options
L7 HTTPS Cluster (172.16.0.203:443)
A default HTTPS cluster with SSL Offloading. Uses the same server pool as the clusters above.
Cluster Options
L7 HTTP Cluster (172.16.0.203:80)
Responder
A default HTTP cluster with a direct responder to reply to clients that mistakenly specify http://

The procedure below shows you how to use one line commands in the global context to set up the configuration illustrated above.

1. Power on Equalizer and enter the CLI, as shown in "Starting the CLI" on page 142.
2. Configure a VLAN for the GUI, SSH, and cluster IP addresses using the format:

   ```
   eqcli > vlan vlname vid vlan_ID
   ```

   Replace ***vlname*** with the VLAN name and ***vlan_ID*** with the VLAN ID number (1-4094). If you are using untagged VLANs (common in many sites), the VLAN ID can be any number not used on another VLAN. If you are using tagged VLANs, ask your network Administrator for the correct VLAN ID to specify.

```
   eqcli > vlan 172net vid 2
```

3. Create a VLAN for servers on the remaining ports:

```
   eqcli > vlan 192net vid 3
```

4. Add subnets to both of the VLANs. You’ll need to specify the subnet IP address, which is the load balancer's address on this network. It must be an IPv4 or IPv6 address in CIDR format (e.g., 172.16.0.200/21).

   Enter the following command syntax, all on one line:

```
eqcli > vlan vlname subnet subnetname ip cidr_ip
```

   In the command above, ***vlname*** is the VLAN name, ***netname*** is the name of the subnet, ***cidr_ip*** is the CIDR format IP address. For example:

```
eqcli > vlan 172net subnet sn01 ip 172.16.0.200/21
eqcli > vlan 192net subnet sn01 ip 192.168.0.200/21
```

5. Configure services on each subnet of each VLAN. For our example, we’ll enable SSH login and the GUI over HTTP on the **172net** VLAN.

```
eqcli > vlan 172net subnet sn01 services http,https,ssh
eqcli > vlan 192net subnet sn01 services http,https,ssh
```

6. Configure routing on the VLANs including a default route and gateway routes. In the example below, `0/0` is the default route and **`172.16.0.1`** and **`192.168.0.1`** are the gateways, which are unadorned IP addresses. In this scenario, all packets for destinations are to be sent via these routes.

```
eqcli > vlan 172net subnet sn01 route 0/0 gw 172.16.0.1
eqcli > vlan 192net subnet sn01 route 0/0 gw 192.168.0.1
```

7. Associate an interface instance with the VLAN. In the example below we assume that you are using the port labeled "1" on the front panel. Enter one of the following commands, depending on whether the VLAN you created above is untagged or tagged (ask your network administrator if you are unsure):

   `eqcli >`**`vlan `*`vlname`*` ifi if01 type untagged`**

   or

   `eqcli > `**`vlan `*`vlname`*` ifi if01 type tagged`**

8. Connect Equalizer to your network on the VLANs that you set up in the previous steps, using the appropriate front panel ports. You should now be able to ping Equalizer’s IP address on each VLAN. If it does not respond on a VLAN, you may need special routes on the default router, or on the next-hop gateway for a particular VLAN.

9. Set the timezone. Enter:

```
eqcli > timezone?
```

10. Locate your timezone in the displayed list and press **`q`** to quit out of the list. Then, type in your timezone number and press <**`Enter`**>, as in this example for the "America/New York’" time zone:

```
eqcli > timezone 161
```

11. If Equalizer can reach the Internet, add a name server so that NTP will work and time will be the same across all Equalizers:

```
   eqcli > name-server IP_address
```

    Otherwise, set the time manually on all systems to the current time:

```
   eqcli > date HHmmss
```

12. Create two real servers:

```
eqcli > server sv01 proto tcp ip 192.168.0.5 port 80
eqcli > server sv02 proto tcp ip 192.168.0.6 port 80
```

13. Create a server pool:

```
eqcli > srvpool sp01 policy adaptive respv 3
```

14. In server pool **sp01**, create server instances for the servers created in Step 6.

```
eqcli > srvpool sp01 si sv01 weight 100
eqcli > srvpool sp01 si sv02 weight 100
```

15. Create a Layer 7 HTTP cluster:

```
eqcli > cluster cl01 proto http ip 172.16.0.201 port 80 srvpool sp01
```

16. Create a Layer 4 TCP cluster using server pool **sp01**, with DSR enabled:

```
eqcli > cluster cl02 proto tcp ip 172.16.0.202 port 80 srvpool sp01 flags dsr
```

17. Add an SSL certificate store (for the HTTPS cluster we'll create later). Enter:

```
eqcli > certificate ct01
```

18. Import the certificate and its associated private key using either of the following methods:

    If the certificate resides on an FTP site, enter commands like the following, substituting the IP address and path on your FTP site from which the certificate and private key can be downloaded:

```
eqcli > certificate ct01
eqcli-cert> certfile ftp://10.0.0.21/certfile.pem
eqcli-cert> keyfile ftp://10.0.0.21/keyfile.pem
```

    If you want to cut and paste the certificate and key using an editor, use commands like the following:

```
eqcli > certificate ct01 certfile edit
eqcli > certificate ct01 keyfile edit
```

    Certificates and keys must be downloaded separately, in PEM format. If a chain of certificates and keys must be uploaded, ensure that all the certificates are in one file and all the private keys are in another.

19. Create a Layer 7 HTTPS cluster using server pool **sp02** and associate certificate **ct01** with the cluster:

```
eqcli > cluster cl03 proto https ip 172.16.0.203 port 443 srvpool sp01
certificate ct01
```

20. Create a Layer 7 HTTP cluster -- do not specify a server pool, since this cluster will be used only to redirect clients to **cl03**:

```
eqcli > cluster cl04 proto http ip 172.16.0.203 port 80
```

21. Add a "sorry" responder that will be used to display a web page that asks the user to try again later:

```
eqcli > resp Sorrycl01 type sorry html edit
```

    An editor is launched so that you can enter the HTML for the responder page. For example, you can enter Once you are done, type **<Esc><Enter>** and then **<Enter>** to save the HTML you entered.

22. Add the responder created in the previous step to cluster **cl01**:

```
eqcli > cluster cl01 resp Sorrycl01
```

The effect of adding this responder to **cl01** is that if all the servers in server pool **sp01** are unavailable, clients making requests to cluster **cl01** will receive an automatic response asking them to try again later.

23. Add a redirect responder that will redirect all requests coming into the same cluster IP as **cl03** on port 80 (via HTTP); the responder will be configured to redirect these requests to **cl03** on port 443 (via HTTPS).

Since some of the arguments to this command are longer than one line, we'll add the responder using multiple command lines to make the input clearer:

```
eqcli > resp Redircl04
eqcli rsp-Red*> type redirect
eqcli rsp-Red*> statcode 301
eqcli rsp-Red*> statdesc "Moved Permanently"
eqcli rsp-Red*> regex "http://clustercl03.example.com/([^ \r]+)?"
eqcli rsp-Red*> url "https://clustercl03.example.com/$1"
eqcli rsp-Red*> exit

eqcli: 12200287: Operation successful
eqcli >
```

Note the following:

The regular expression used in the **regex** parameter contains a single space between the caret (**^**) and backslash (****) characters.

The FQDN used in the **regex** and **url** parameters (e.g., **cluster-cl03.example.com**) must match the FQDN used by clients to connect to cluster **cl03**.

24. Add the responder created in the previous step to cluster **cl01**:

```
eqcli > cluster cl04 resp Redircl04
```

Since **cl04** has no associated server pool specified in its configuration, all requests coming in to **cl04** will be redirected to **cl03** by the responder.

Chapter 10

Upgrading and Downgrading

Sections within this chapter include:

Upgrade Path from EQ/OS 8.6 to the Latest EQ/OS 10 Version

You can upgrade directly from EQ/OS 8.6.0i to the latest version of EQ/OS 10.

Downgrade Path

The table below shows the configuration file versions and their associated EQ/OS 10 releases. Configuration file version 6 is the latest configuration file used. When you downgrade, you must downgrade one configuration version at a time. For example, if you want to downgrade from EQ/OS 10.3.1a to EQ/OS 10.1.0a, you must use the following downgrade sequence: **10.3.0a > 10.2.0a > 10.1.0a**.

Configuration File Version	Equalizer LX/GX Releases
Version 1	All releases prior to 10.1.0a
Version 2	10.1.0a
Version 3	10.2.0a
Version 4	10.3.0a
Version 5	10.3.0e

EQ/OS 8.6 Upgrade Procedure

To upgrade your EQ/OS 8.6 Equalizer to EQ/OS 10:

1. Connect Equalizer with a serial console. Refer to "Setting Up a Terminal or Terminal Emulator" on page 46 .
2. Set up a local FTP server that can be accessed by Equalizer. This will be used during the upgrade process to save a EQ/OS 8.6 system image that can be used to restore Equalizer to EQ/OS 8.6. The creation of the restore image is required in order to be able to downgrade Equalizer back to EQ/OS 8.6.

Important - The EQ/OS 8.6 system must have the latest release of EQ/OS 8.6 on both disk partitions, running the same configuration. To accomplish this, do one of the following:

- **If you are already running the latest EQ/OS 8.6 release, upgrade to the same version of EQ/OS 8.6 again.**
- **If you are not running the latest EQ/OS 8.6 release, upgrade to the latest EQ/OS 8.6 release, reboot, and then upgrade to the latest 8.6 release again.**

3. **Equalizer must be running from the first disk partition when you begin the upgrade.** The easiest way to ensure this is to reboot the unit and press **F1** (the first function key on the keyboard) when the following prompt appears:

```
F1  FreeBSD
F2  FreeBSD
Default: F1
```

The "`Default:`" line above may contain "`F1`" or "`F2`". Press **F1** regardless of what is displayed on the "`Default:`" line.]

4. Log into the EQ/OS 8.6 console as **root** using the serial port on Equalizer’s front panel. [An upgrade to EQ/OS 10 cannot be performed using the EQ/OS 8.6 GUI, nor can it be performed over an SSH connection. You must login using the serial console interface.]
5. At the system prompt, enter:

```
eqadmin
```

The **eqadmin** interface menu is displayed.

6. Select option "**8 Upgrade**" by pressing the "**8**" key followed by "**Enter**" on the console keyboard.

7. Do *one* of the following:

- Press "**1**" followed by "**Enter**" to download the upgrade image from the Coyote Point web site.
- Press "**2**" followed by "**Enter**" to download the upgrade image from a local server.

8. Enter the upgrade URL using the EQ/OS 8.6 syntax and press "**Enter**". For example, the following URL downloads the image from a local server:

```
ftp://10.0.0.121/pub/patches/upgrades/10.0.0/os8upgrade/upgrade.tgz
```

9. Once the upgrade image is downloaded, the following prompt is displayed:

```
Equalizer OS 8.6 -> OS 10.x UPGRADE SCRIPT

This script will COMPLETELY REMOVE your existing Equalizer configuration
and replace it with a blank (unconfigured) installation of EQ/OS version
10.

If you proceed further, load balancing will be disabled on this
Equalizer until it is next rebooted, even if the upgrade fails.

Continue with upgrade [Y/N]?
```

Press "**Y**" and then "**Enter**".

Note - You may see the following messages during an upgrade:
tar: Unable to access licenses (No such file or directory)
tar: WARNING! These file names were not selected:licenses

The 'licenses' directory is an artifact of earlier releases and is no longer needed (licenses are now stored in the configuration file). The upgrade script, however, still expects it to be there; this will be fixed in a future release. This issue is benign and does not indicate any problem with Equalizer upgrade, licensing,configuration, or behavior.

10. The following prompt is displayed:

```
Requesting pid 460 to terminate.

It is safest to proceed only on a system with an active support
contract. It is recommended that you verify that your support
status by re-licensing this system using the online license
server now.

If this is not possible because your system does not have
Internet access, please contact Coyote Point or your Coyote
Point distributor to confirm active support status.

This Equalizer has serial number A09BA-16003.

Re-license system now [Y/N]?
```

Do *one* of the following:

If your network configuration will not allow Equalizer to contact the Coyote Point license server over an internet connection, please call Coyote Point Support or your local distributor to confirm your support status. Then, press "**N**" and "**Enter**" to proceed with the upgrade. Two confirmation messages are displayed; to proceed, press "**Y**" and then "**Enter**" at each prompt to proceed with the upgrade.

Otherwise, press "**Y**" and then "**Enter**" to request a new license. If successful, the following message is displayed and contains Equalizer’s serial number:

```
Successfully re-licensed e450gx-1 serial_number.
```

11. The following message is displayed:

```
PERMANENTLY upgrade this system to EQ/OS 10 [Y/N]?
```

Press "**Y**" and then "**Enter**" to proceed with the upgrade.

12. The user is now prompted for the second stage upgrade URL:

```
This installer uses a 2-stage process. You must enter the URL for
the second-stage install bundle. It is usually the same URL from
which you retrieved the first-stage installer, except without the
last component.

For example, if you retrieved this installer from:
http://www.coyotepoint.com/example/upgrade.tgz

Then the second-stage install bundle URL would be:
http://www.coyotepoint.com/example/

Enter the URL:
```

Enter the second stage URL and press "**Enter**" to continue with the upgrade. In our example, the second stage URL would be:

```
ftp://10.0.0.121/pub/patches/upgrades/10.0.0/os8upgrade
```

13. After the second stage upgrade files are downloaded, verified, and unpacked, you are asked to create a restore image:

```
Upgrade bundle is EQ/OS 10.0.2a.

Checking that bundle is EQ/OS 8 to EQ/OS 10 type.
Retrieving autobuilds/folsomBuilds/18288/i386/binary/os8upgrade//is-os8
100% 11 00:00 ETA
It is very important to create a restore image of this Equalizer
running the current EQ/OS 8 software. This is not a standard
backup of the EQ/OS 8 system but an image which will allow you
to downgrade this particular system from EQ/OS 10 should you
so desire.

Without a restore image, this system can not be downgraded to
EQ/OS 8.

The restore image will require approximately 200MB of free space
on an FTP server to which you can upload files.

Do you want to create a restore image [Y/N]?
```

Press "**Y**" and then "**Enter**" to create a restore image.

14. The system then prompts you to enter a URL for the restore image as well as a username and password to :

```
Do you want to create a restore image [Y/N]? ^Cy
Cleaning up log and temporary files before restore imaging process.
Flushing disk write cache.
Building restore image.
3891+0 records in.1 MiB / 242.1 MiB = 0.348 1.2 MiB/s 3:23
3891+0 records out
255000576 bytes transferred in 204.873566 secs (1244673 bytes/sec)
100 % 85.3 MiB / 243.2 MiB = 0.351 1.2 MiB/s 3:24
Taking fingerprint of restore image.
Sending restore image to remote FTP server.
Enter URL for path to which to send restore image.
Example: ftp://ftp.coyotepoint.com/my_images/
```

15. You are now prompted for an FTP server to which the encrypted restore image can be transferred:

```
Sending restore image to remote FTP server.
Enter URL for path to which to send restore image.
Example: ftp://ftp.coyotepoint.com/my_images/

Enter URL:
Enter Username for file upload:
Enter Password for file upload:
```

    Enter the **URL**, and then a **Username** and **Password** to log into the FTP server.

16. After you supply the FTP login information, Equalizer uploads the image to the FTP server, and then downloads it to verify that it is correct. If the verification succeeds, Equalizer continues with the upgrade and reboots automatically after displaying this prompt:

```
Activating second stage install partition for next reboot.
rebooting -- this may take up to a full minute.
```

17. After rebooting, the system will automatically continue the upgrade by booting from the second partition. DO NOT PRESS ANY KEYS WHEN THE BOOT MENU IS DISPLAYED. Wait for the system to boot automatically. Once the system boots, it unpacks the EQ/OS 10 upgrade image and creates the appropriate file systems. When it is done, the following prompts are displayed:

```
The operating system has halted.
Please press any key to reboot.
```

18. Press any key to reboot the system.
19. As the system reboots, you may see prompts indicating that the front panel switch firmware needs to be upgraded:

```
Switch firmware is down-level.
WARNING: This upgrade contains firmware which requires
an immediate reboot after installation, which will be
automatically performed.
```

    The switch firmware is automatically upgraded if required. This process can take several minutes.

Caution - DO NOT INTERRUPT THE SWITCH FIRMWARE UPGRADE IN ANY WAY. This includes power cycling the unit. Interrupting the switch firmware upgrade can leave your system in an inoperable state.

- After successfully upgrading and validating the switch firmware, the system reboots automatically. Once the system finishes rebooting, the console displays the EQ/OS 10 CLI "Username:" prompt. You can now log into the CLI and configure Equalizer into your network.

  See "Basic Configuration" on page 50 for the basics of creating a VLAN over which you can access the GUI and use SSH to log into the console.

  See "Sample Configuration" on page 68 for a brief tutorial that explains how to perform additional configuration tasks (creating servers, clusters, etc.) using the CLI.

Downgrading to EQ/OS 8.6

If you upgraded Equalizer from EQ/OS 8.6 to EQ/OS 10, you can later downgrade Equalizer back to the release that was running when you upgraded. You can downgrade any Equalizer in the GX series.

You cannot downgrade LX series Equalizers.

This procedure requires the following:

- A saved restore system image created during the upgrade to EQ/OS 10.
- The password for the saved system image.

If you did not record the password or have lost it, contact Coyote Point Support for help.

Note - The downgrade process requires a restore image created during the upgrade of the unit from EQ/OS 8.6 to EQ/OS 10. If you later choose to downgrade a EQ/OS 10 Equalizer back to EQ/OS 8.6, you must use a restore image that was created during the upgrade of that unit to EQ/OS 10. The file name used to save a restore image is the serial number of the unit on which it was created, and this restore image can only be used to restore the Equalizer with the same serial number. On EQ/OS 10, you can use the CLI command to display Equalizer’s serial number, or check the serial number tag on Equalizer’s back panel.

1. Connect Equalizer with a serial console. Refer to "Setting Up a Terminal or Terminal Emulator" on page 46.
2. Log into the CLI.
3. At the global context prompt, enter:

```
eqcli > upgrade URL
```

The URL is an unadorned `ftp://` or `http://` URL that completely specifies the path to the downgrade directory. In the example shown, replace the current version on your ADC, which can be found on the EQ/OS 10 Support Page.

```
eqcli > upgrade ftp://ftp.coyotepoint.com/pub/patches/upgrades/current
version/os8downgrade
```

4. The downgrade software is downloaded, unpacked, and run. The following prompt is displayed:

```
Please enter the URL for the the system restore image for THIS SYSTEM.

If a username and/or password is required in order to retrieve the
file, the username (and optionally the password) must be embedded in
the URL using standard URL syntax.

For example, if the file is at:
ftp://www.coyotepoint.com/example/Z20CA-31337.xrb

And you use username 'user' and password 'pass' to access that site,
then use URL:
ftp://user@www.coyotepoint.com/example/Z20CA-31337.xrb
or URL:
ftp://user:pass@www.coyotepoint.com/example/Z20CA-31337.xrb
to retrieve the file.

Please enter the URL:
```

Enter the URL where you saved the restore system image created during the upgrade to EQ/OS 10, as in this example:

```
ftp://ftp@10.0.0.21/folsom/A107A-17004.xrb
```

5. The downgrade script then retrieves, decrypts, and installs the restore image from the URL you provided. During this process you are asked to enter the restore image password. At the prompts indicated in the sample output below, enter the `restore image` password (`restore_password`) and press the Enter key to continue:

```
Retrieving expected SHA1 signature for restore image file. Computing SHA1
signature of restore image file.
If you were prompted (and re-prompted) to enter a restore image
password when you created the restore image, then the image was
encrypted.
Was the restore image encrypted [Y/N]?
```

If the restored image was originally encrypted with a password, you will be prompted with the following after selecting "Y".

```
Enter your restore image password: restore_password
Enter your restore image password again: restore_password
Password: restore_password
```

In either case the following will be displayed as the system restores the image.

```
Formatting target filesystem.
/dev/rwd0a: 207.4MB (424680 sectors) block size 8192, fragment size 1024
using 5 cylinder groups of 41.48MB, 5309 blks, 10304 inodes.
32,
84976,
169920,
254864,
339808,
Installing restore environment onto target filesystem.
Decrypting image restore data onto target filesystem.
Password: restore_password
Writing secondary boot configuration.
Updating primary bootblock version table.
Performing automatic reboot. System will reboot to
image extraction environment.
Halt NOW!
```

6. As the system reboots, *do not press any keys*. After the following prompt is displayed:

```
IMAGE RESTORE DOWNGRADE IN PROGRESS, DO NOT CUT SYSTEM POWER
```

The system boots and copies the restore image onto disk:

```
Beginning image restore process.
/tmp/restore.img.xz (1/1)
```

Once the image is restored, the system reboots again. After the reboot is complete, the EQ/OS 8.6 login prompt is displayed.

Upgrading to the Latest Release

Upgrade Path

The current Equalizer uses configuration file version 6. You can upgrade to any higher version of the OS regardless of the version you are currently running.

Upgrade using the CLI

To upgrade a system that is already running EQ/OS 10 to the latest release using the CLI, do the following:

1. Ensure that the upgrade image is available on an FTP or HTTP server that is accessible to Equalizer. This can be either the Coyote Point Systems, A Subsidiary of Fortinet, Inc. Upgrade server or a local server.
2. Log in to the Equalizer CLI using the serial console or via SSH on a VLAN that is configured for SSH access.
3. At the `eqcli >`prompt, enter:

```
eqcli > upgrade URL
```

The URL is an unadorned `ftp://` or `http://` URL that completely specifies the path to the upgrade image *directory*, as in this example:

```
eqcli > upgrade ftp://X.X.X.X/pub/patches/upgrades/10.X.X/upgrade
```

4. Equalizer downloads the upgrade files automatically, unpacks them, and then begins the upgrade. No user intervention is required.

   When the upgrade is complete, the following messages are displayed:

```
Upgrade successfully completed. New version is default at next system boot.
```

5. To reboot the system and run the newly installed version, enter:

```
eqcli > reboot
```

Upgrade using the GUI

Refer to "Manage Software" on page 292 for instructions on upgrading using the GUI.

Chapter 11

Load Balancing & Networking

Sections in this chapter include:

Networking Technologies

There are several networking technologies described herein that apply to Equalizer installations. They are summarized below, however, specific rules and commands are described further as each networking scenario is described in detail.

Destination Routing

This is standard routing that is performed by any networking device. The device determines how to send a packet to its destination by evaluating the destination IP address to see if it is on a local network. It if is, the device sends the packet directly using the Ethernet layer. If, however, that IP address is on a remote network, the device evaluates its routing table to determine how to send it. The routing table consists of a set of entries in the form:

```
IP/NETMASK || GATEWAY
```

The device searches the routing table in a <u>most</u> specific to <u>least</u> specific manner in order to find the most appropriate route to use. For example, if one entry is for the network 10.0.0.0/8 and another is for the network 10.0.0.0/24, a packet destined for the IP address 10.0.0.1 would use the /24 entry because it is more specific. However, a packet destined for 10.0.1.1 would use the /8 entry because the /24 entry does not apply to this destination. Once a matching route is found, the device sends the packet on to the gateway (or router) that is specified in this route. It is then this gateway's job to get the packet closer to its final destination.

Source-Based Routing

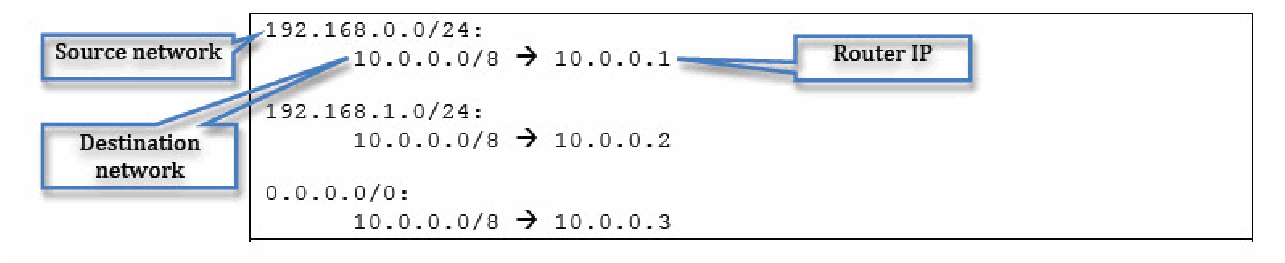
This concept is not unique to Equalizer, however the behavior of each device that implements Source-based Routing can be different. The definition of source-based routing is simply that the source IP address is used in the routing decision. For Equalizer, this means that rather than having a single destination routing table, the system actually has a set of destination routing tables, each used only when the source IP address of a packet matches a particular network. A source-based routing table contains entries in the form:

```
(SOURCE IP/NETMASK,DESTINATION IP/NETMASK) || GATEWAY
```

If the destination IP address is on a local network, source-based routing is <u>not</u> used. The packet is sent to the destination system via Ethernet.

If the destination IP address is on a local network, source-based routing is not used by default. The packet is sent to the destination system via Ethernet. However, administrators can configure their routing tables to override the local entries for particular networks, in which case Equalizer will prefer a source route over a local network route. If configured in this manner, Equalizer will send the packet to an IP gateway associated with the source route rather than simply using the ARP address of the destination system to send the packet over Ethernet directly.

If the destination IP address is on a remote network, the device trying to send a packet performs a most-specific to least-specific search for the source IP network. If a matching source route is found within the routing table, any routing table entries that contain that source IP network are used as a destination routing table. For example:



```
192.168.0.0/24:
        10.0.0.0/8 → 10.0.0.1

192.168.1.0/24:
        10.0.0.0/8 → 10.0.0.2

0.0.0.0/0:
        10.0.0.0/8 → 10.0.0.3
```

This source-based routing table shows three source routes. If a packet sent to 10.0.0.100 is coming from the 192.168.0.0/24 network, Equalizer will use 10.0.0.1 as the gateway. If it is coming from the 192.168.1.0/24 network, Equalizer will use 10.0.0.2 as the gateway. If it is coming from any other network, Equalizer will use 10.0.0.3 as the gateway.

The IP 0.0.0.0/0 is the least specific that a network entry can be - it matches every IP address. However, because of the most-specific to least-specific search that Equalizer performs, the 0.0.0.0/0 source route is not used unless none of the other routes match.

Also note that in this configuration, any packets that have a destination IP address other than a network local to Equalizer (presumably 192.168.0.0/24, 192.168.1.0/24 and 10.0.0.0/8), a route would not be found and the packet would be dropped by the system. To prevent this from happening, most configurations include a default route in the form (0.0.0.0/0, 0.0.0.0/0) || GATEWAY.

Local Networks

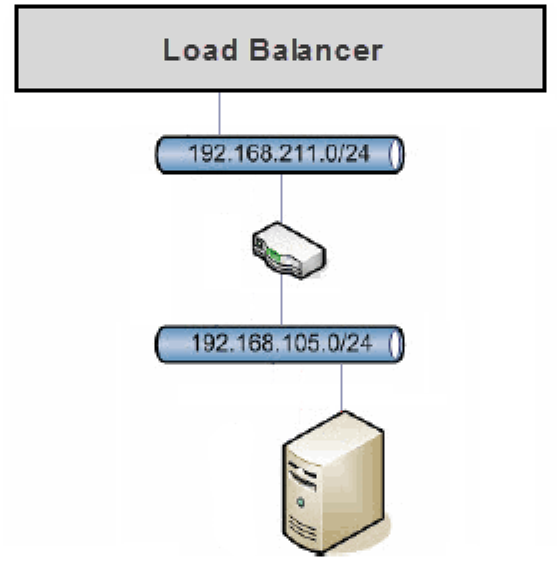
Any network that has been added as an Equalizer subnet is considered local to Equalizer. When a subnet is configured, an Administrator assigns an IP address (potentially more than one) that is Equalizer's IP presence on this subnet. When an Equalizer is referred to as being in *single-network* mode or *dual-network* mode, this is the number of local networks.

Remote Networks

Any network that is not a local network. This means that Equalizer needs to perform routing to communicate with a device on this network.

Destination Networks

A specific remote network that has been configured by the Administrator as connected to a local network of Equalizer. This means that if Equalizer needs to send packets to this network, it should do so from an IP address on the local network and use the router of the local network. For example:

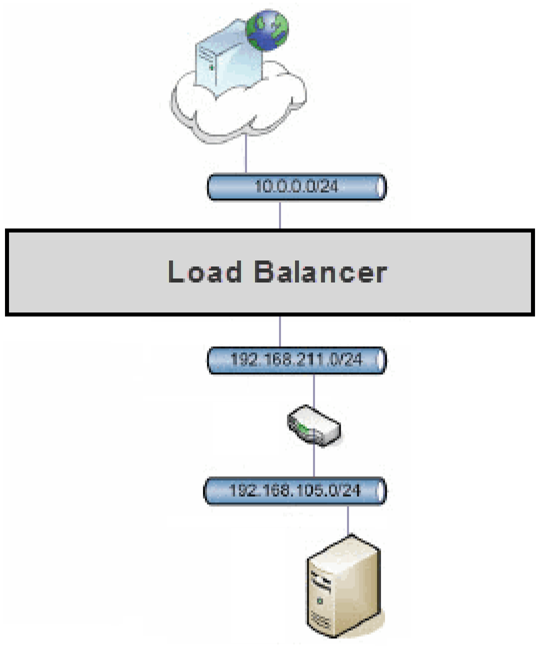


In this configuration, 192.168.211.0/24 is a local network for Equalizer, configured by adding a subnet to the configuration. 192.168.105.0/24 can be configured as a destination network of the 192.168.211.0/24 network. When adding a destination network, the administrator is configuring several things:

- In order to send packets from Equalizer to the destination network, Equalizer should use its IP address on the local network. This how Equalizer selects an IP address to use when sending a packet to the destination network. In order to do this, Equalizer actually sorts all of the destination networks it knows about in most-specific to least-specific order. It then chooses an appropriate IP address to use based on the first destination network to match.
- Normally, Equalizer would not allow any packets that do not have a source IP address on the local network. Adding the destination network means that Equalizer will now allow packets from this network to be routed with the same rules as packets from the local network.
- Similarly, Equalizer will automatically add source routes for packets from the destination network that match existing source routes for the local network.
- If outbound NAT has been configured for the local network, analogous rules are added for the destination network.

Outbound NAT

NAT, or Network Address Translation, is a common concept for most network administrators. Equalizer administrators usually need to enable NAT when a server on an "internal" (non-public, DMZ) network needs to access resources on the Internet or another public network. This internal network can be either a local network or a destination network for Equalizer. In this scenario, the administrator enables outbound NAT and selects the local network that should be used to NAT packets from the internal network. For example:



In this example, neither the 192.168.211.0/24 nor the 192.168.105.0/24 networks can access the Internet directly. The administrator configures Equalizer to provide outbound NAT service for these networks by using an IP address on the 10.0.0.0/24 network when these internal networks need to talk to the Internet.

When configuring outbound NAT, the internal local network that is being configured for outbound NAT must use the routing information for the external network which it is using NAT through. In the example above, the default gateway for the 192.168.211.0/24 network will really be on the 10.0.0.0/24 network.

This is logical when you remember it this way: If Equalizer is sending a packet from the 192.168.211.0/24 network to a host on the Internet, it has to be sent through the gateway of the external network, rather than the internal network.

When Outbound NAT is enabled for a local network that contains attached destination networks, the destination networks automatically inherit the same outbound NAT configuration.

Note - Outbound NAT is not supported for IPv6.

Network Permissions

Local networks configured in Equalizer use a default/deny permission scheme. This means that if an Administrator wants to route between two networks using Equalizer, they must explicitly enable permissions between that pair of networks.

Note that permissions are not symmetrical: it is possible to configure a solution where one network can talk to another but not vice-versa. For most configurations, permissions are necessary on both networks: if network "A" needs to route to network "B", a permission must be added to "A" for "B" and another permission must be added to "B" for "A".

Permissions are only necessary when using Equalizer to route packets. They are not required for Application Traffic Management. That is, when an Equalizer cluster is paired with a server (by adding a server pool containing that server to the cluster), Equalizer knows that any packets associated with a connection for that cluster should be allowed on the server network.

Networking Conventions

Several conventions are used within this section:

- Network addresses are represented in Classless Inter-Domain Routing (CIDR) notation, an IP addressing scheme in the form A.B.C.D/X where X is the number of bits in the subnet mask.
- Subnets are referenced by the name of the VLAN which contains them, followed by the subnet name. For example, internal:net means VLAN internal, subnet net.
- All VLAN configurations presented are untagged. The configurations and concepts in this document applies for tagged VLANs as well.
- This section uses examples that are for IPv4 networking. However, the configuration for IPv6 networking would be identical- with a couple of exceptions. These exceptions are identified - where applicable.
- This section uses examples from an Equalizer OnDemand system using untagged VLANs. If your configuration uses tagged networks or Equalizer physical appliances, the network interfaces displayed here will not match your configuration. This is normal and remainder of the section still applies

Common Equalizer Networking Scenarios

This section describes individual networking scenarios that can be used to build up a large, more complicated configuration for Equalizer . Each section starts at a specific pre-configured configuration, and references the section which helps set up that configuration.

Blank Configuration

When the Equalizer configuration does not contain any subnets, the networking configuration should also be blank:

```
eqcli > show sbr

IPv4 Default Source Selection Table:
                0/0 172.16.5.90

IPv6 Default Source Selection Table:

Source Routing Table:

        0.0.0.0/00:
                0.0.0.0/00 via 172.16.0.1 weight 0 (!prefer)

        172.16.0.0/21:
                0.0.0.0/00 via 172.16.0.1 weight 0 (!prefer)

IP Filter Rules:
empty list

IP NAT Rules:

List of active MAP/Redirect filters:

List of active sessions:

eqcli >
```

Single VLAN/Subnet

A Single VLAN/subnet configuration is one of the most common scenarios used. In this setup, Equalizer is placed into an existing network so that all servers, internal clients, and external routers are on the same VLAN. (This usually means special routing on the servers or the use of **no spoof** for Equalizer clusters. See "Overview" on page 324.)

Here a single VLAN is added , and a subnet is configured on the VLAN:

```
eqcli > vlan internal vid 1
eqcli: 12000287: Operation successful

eqcli > vlan internal subnet net ip 192.168.211.8/24
eqcli: 12000287: Operation successful
```

There are no differences to the DSS (Default Source Selection table), which is a listing of all destination networks configured in the load balancer), the routing, and the NAT tables, since no new entries have been added to them. However, the IP Filters table has been updated by the system:

```
IP Filter Rules:

IPv4 Rules:
1: pass on interface lo0 all hits: 0 bytes: 0

2: pass on interface wm1 hits: 227 bytes: 7025
From To
192.168.211.0/24 -> 192.168.211.0/24

3: block all hits: 26 bytes: 2579

IPv6 Rules:
1: pass on interface lo0 all hits: 0 bytes: 0

2: pass hits: 0 bytes: 0
From To
fe80::/10 -> any

3: block all hits: 0 bytes: 0
```

The new rule shows that packets from network internal:net are allowed into the system if they are being sent to the same network. Without this rule, the newly added IP address could not be reached from the rest of the network.

Also note that IPv4/6 rule 1 allows Equalizer traffic if it is on the local host interface (lo0), and IPv4/6 rule 3 blocks all traffic which didn't fall into one of the previous rules. This is the default deny rule. IPv6 rule 2 is an automatically-added rule for link-local IPv6 addresses, which is always there if any networks are configured.

If all of the clients and servers for this Equalizer are on the internal:net network, we're done, however, most installations have customers which are on a different network, usually the Internet.

Single VLAN/Subnet with a Default Gateway

A system can be connected to the Internet by adding a default route (the newly-added rules are in italics) because there is only a single Equalizer local network.

```
eqcli > vlan internal subnet net default_route 192.168.211.1
eqcli: 12000287: Operation successful
```

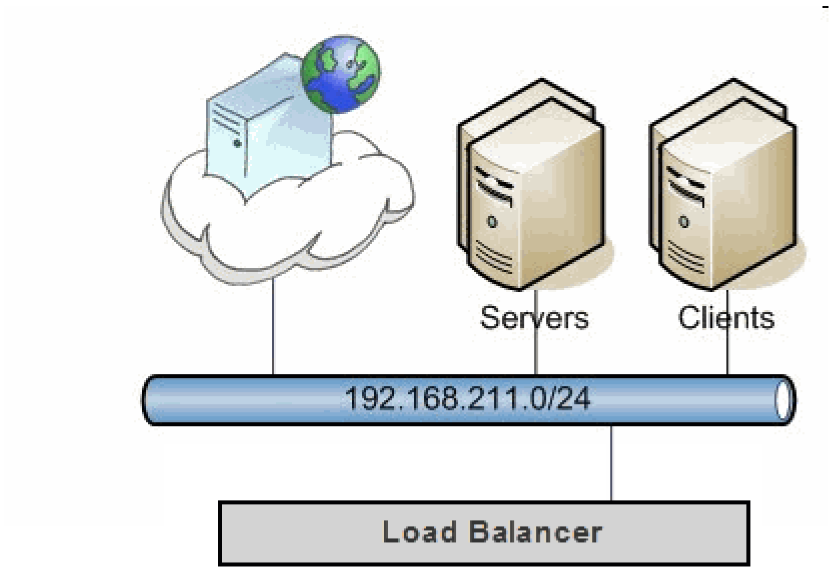
```
Source Routing Table:
192.168.211.0/24:
        default         via 192.168.211.1

IP Filter Rules:

IPv4 Rules:

1: pass on interface lo0 all hits: 0 bytes: 0

2: pass on interface wm1 hits: 32 bytes: 1368
                From            To
        192.168.211.0/24 -> 192.168.211.0/24

3: block on interface wm1 hits: 0 bytes: 0
                From            To
        192.168.211.0/24 -> 192.168.211.0/24

4: pass on interface wm1 hits: 0 bytes: 0
                From            To
        192.168.211.0/24 ->     any

5: pass on interface wm1 hits: 0 bytes: 0
                From            To
                any ->     192.168.211.0/24

6: block all hits: 7 bytes: 799

IPv6 Rules:
1: pass on interface lo0 all hits: 0 bytes: 0

2: pass hits: 0 bytes: 0
                From            To
                fe80::/10  ->   any
```

Now that we have a non-blank routing configuration, we can see that the source routing table reflects the change, and that we have a couple of routing-specific IP Filter rules:

Rule 3 is inserted immediately after any 'pass' rules for this subnet. Because there aren't any other subnets except this one, this rule will not be used (the previous rule allows all packets that this rule would block).

Rules 4 and 5 allow traffic from non-Equalizer networks into Equalizer and from Equalizer to non-Equalizer networks. These are the rules that allow routing through the default gateway to work.

The configuration presented in this section corresponds to the following scenario:



Dual VLAN/Network

Another typical configuration is to have two networks connected to Equalizer:

1. One for external connectivity (this is where the Equalizerclients and clusters are)
2. One for internal resources (this is where the servers are)

We start with a single-VLAN configuration with no default route (See "Single VLAN/Subnet" on page 94) and add a second network for external connectivity, along with a default route for that network, as shown below.

```
eqcli > vlan external untagged_ports 1 vid 2
eqcli: 12000287: Operation successful

eqcli > vlan external subnet net ip 10.0.0.68/24 default_route 10.0.0.254
eqcli: 12000287: Operation successful
```

The IP Filter configuration is updated as shown below:

```
Source Routing Table:
        10.0.0.0/24:
                default         via 10.0.0.254

IP Filter Rules:

IPv4 Rules:
1: pass on interface lo0 all hits: 0 bytes: 0

2: pass on interface wm1 hits: 36 bytes: 1608
                From            To
        192.168.211.0/24  ->  192.168.211.0/24

3: pass on interface wm0 hits: 48 bytes: 2926
                From            To
        10.0.0.0/24  ->  10.0.0.0/24

4: block on interface wm0 hits: 0 bytes: 0
                From            To

5: pass on interface wm0 hits: 27 bytes: 4916
                From            To
        10.0.0.0/24  ->         any

6: pass on interface wm0 hits: 0 bytes: 0
                From            To
                any  ->  10.0.0.0/24

7: block all hits: 1 bytes: 328
```

The 192.168.211.0 network rules remain unchanged. We have new rules for the 10.0.0.0 network:

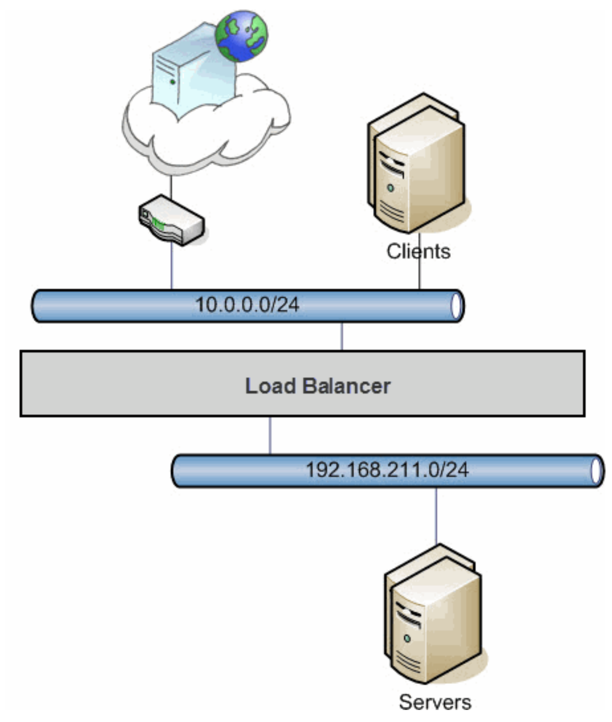
Rule 3 is for sending packets on the external network interface (wm0 in this case) to the 10.0.0.0 network from the 10.0.0.0 network.

Rules 5 and 6 for packets between the 10.0.0.0 network to any other network.

Note that *Rule 4* is a block rule which prevents traffic between the 10.0.0.0 network and all subnets known to the system. Such a rule doesn't exist for the 192.168.211.0 network because we have not enabled routing for it.

Since the new *external* network is the one is used for sending packets to the Internet, we also make it the default network for sourcing packets.

We see that setting this flag has created a DSS table entry. This entry is a definition for the 0/0 destination network, which specifies that the *external* VLAN is the one connected to this network, and when Equalizer needs to send packets to this network, it should use the 10.0.0.68 IP address. This setup is sufficient for most dual-network configurations:

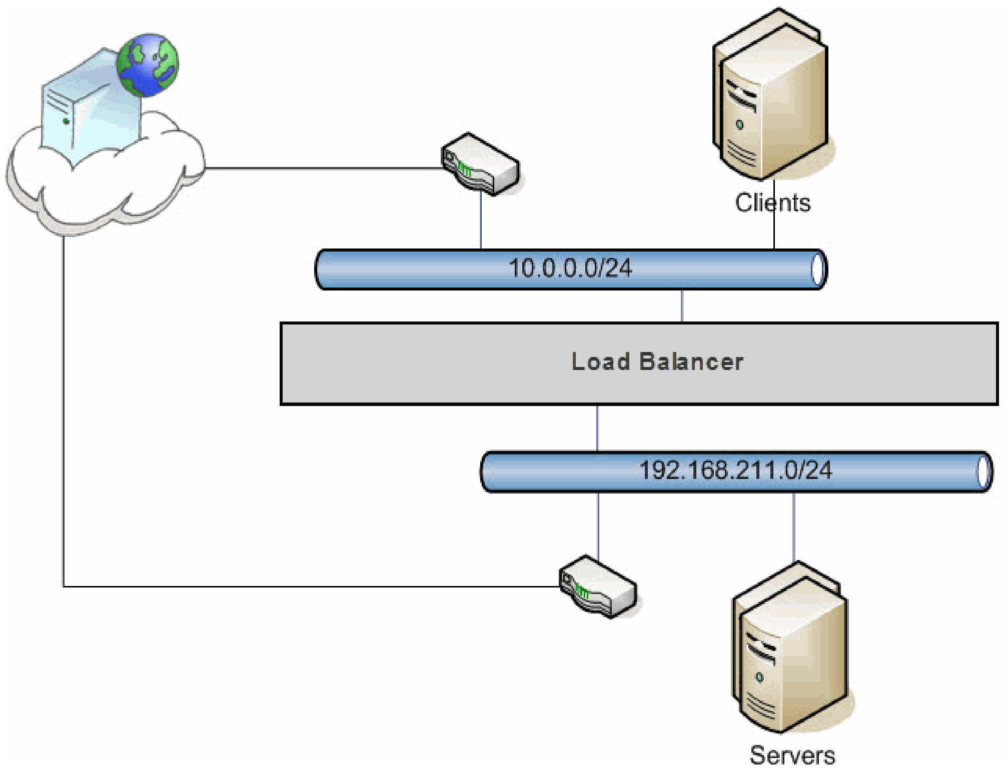


With this configuration, clients can connect to cluster IP addresses on the 10.0.0.0 network, and Equalizer will send the requests to the servers on the 192.168.211.0 network.

```
Source Routing Table:
        0.0.0.0/00:
                default via 10.0.0.254

        10.0.0.0/24:
                default via 10.0.0.254

IP Filter Rules: IPv4 Rules:

1: pass on interface lo0 all hits: 0 bytes: 0

2: pass on interface wm1 hits: 141 bytes: 7025
                From             To
        192.168.211.0/24  ->   192.168.211.0/24

3: pass on interface wm0 hits: 5 bytes: 399
                From             To
        10.0.0.0/24  ->  10.0.0.0/24
          0.0.0.0/0      0.0.0.0/0

4: block on interface wm0 hits: 0 bytes: 0
                From             To
        10.0.0.0/24  ->  192.168.211.0/24
                        10.0.0.0/24
                         0.0.0.0/0

5: pass on interface wm0 hits: 4 bytes: 756
                From             To
         10.0.0.0/24  ->        any

6: pass on interface wm0 hits: 0 bytes: 0
                From             To
                any  ->         10.0.0.0/24
                                0.0.0.0/0

7: block all hits: 0 bytes: 0
```

Dual VLAN/Network with 2 Gateways

Imagine a scenario very similar to the one described in Dual VLAN/Network, but the internal network is also able to route to the Internet:



As far as Equalizer is concerned, the configuration doesn't have to change at all from the previous scenario. There is still a single destination network (the Internet), and Equalizer is statically configured to use the 10.0.0.0 network to communicate with this destination network.

The administrator can set up the servers on the 192.168.211.0 network to use their router when sending packets to the Internet, and to use Equalizer whenever sending packets to clients. However, in order to do this on a server, the administrator would need to statically define which portions of the Internet should use which gateway (the router or the Equalizer). This can be configured very simply on Equalizer, instead:

```
eqcli > vlan internal subnet net default_route 192.168.211.2
eqcli: 12000287: Operation successful
```

This command adds a default route for the internal network that is different than the external default route. This means that any traffic coming from the internal network will be source routed through the 192.168.211.2 gateway, while any other traffic will still be routed through the 10.0.0.254 gateway as configured for the external network.

This can be verified by looking at the `eqcli > show sbr` output:

```
Pv4 Default Source Selection Table:
0.0.0.0/00:
default         via 10.0.0.254

192.168.211.0/24:
default         via 192.168.211.2

10.0.0.0/24:
default         via 10.0.0.254
```

The IP Filter rules are updated as well, analogous to the rules which were created when we added routing in Single VLAN/Subnet with a Default Gateway. The new rules allow routing from the internal network.

```
IPv4 Rules:

1: pass on interface lo0 all hits: 0 bytes: 0

2: pass on interface wm1 hits: 39 bytes: 1368
                From                    To
                192.168.211.0/24        192.168.211.0/24

3: pass on interface wm0 hits: 12 bytes: 624
                From                    To
                10.0.0.0/24             10.0.0.0/24
                                        0.0.0.0/0
                                        0.0.0.0/0

4: block on interface wm1 hits: 0 bytes: 0
                From                    To
                192.168.211.0/24        192.168.211.0/24
                                        10.0.0.0/24
                                        0.0.0.0/0

5: pass on interface wm1 hits: 0 bytes: 0
                From                    To
                192.168.211.0/24        any

6: block on interface wm0 hits: 0 bytes: 0
                From                    To
                10.0.0.0/24             192.168.211.0/24
                                        10.0.0.0/24
0.0.0.0/0
7: pass on interface wm0 hits: 4 bytes: 756
                From                    To
                10.0.0.0/24             any
```

```
8: pass on interface wm1 hits: 0 bytes: 0
                From            To
                any             192.168.211.0/24

9: pass on interface wm0 hits: 0 bytes: 0
                From            To
                any             10.0.0.0/24
                                0.0.0.0/0
10: block all hits: 1 bytes: 328
```

It can also be verified using the traceroute tool, available in most Operating Systems. If a traceroute is performed from the server, a different second-hop gateway is used than the first-hop gateway on the Equalizer traceroute:

```
freebsd# traceroute 64.13.152.126
traceroute to 64.13.152.126 (64.13.152.126), 64 hops max, 40 byte
packets
1 192.168.211.8 (192.168.211.8) 0.576 ms 0.799 ms 0.241 ms
2 192.168.211.2 (192.168.211.2) 0.522 ms 0.547 ms 0.334 ms

Equalizer# traceroute -n 64.13.152.126
traceroute to 64.13.152.126 (64.13.152.126), 64 hops max, 40 byte
packets
1 192.168.8.2 1.653 ms 1.342 ms 1.225 ms
```

In the example above, the server (`freebsd`) uses the Equalizer (192.168.211.8) as its gateway, and the Equalizer sends the packet on the 192.168.211.2 gateway. However, when the Equalizer performs a traceroute to the same location, it uses the 192.168.8.2 gateway.

Dual VLAN/Network with Outbound NAT

If we start with the configuration in Dual VLAN/Network, it should be noted that this configuration is not sufficient if the servers on the internal network require Internet connectivity. Equalizer will properly send traffic from the internal network to the Internet, but because the internal network is non-routable, hosts on the Internet will not be able to respond. One way to solve this problem is to have a separate NAT gateway for the server network, as described in Dual VLAN/Network with 2 Gateways. However, because most locations have a single outbound link, configurations with only a single gateway must use Outbound NAT.

Note - The Outbound NAT feature is not available for IPv6 on Equalizer.

Outbound NAT allows the administrator to associate two subnets together using the outbound_nat parameter. The **`from`** address is the source IP address (or range of addresses) to which this NAT rule applies. Use a CIDR-format IP address to specify a range. If the source IP address of an outbound packet matches this IP address (or falls within the specified range), then the packet is modified to use the IP address specified by the **`out`** parameter as the source IP.

The **`out`** address specifies that if the source IP address of an outbound packet matches the IP address (or IP address range) specified by the **`from`** parameter, then the packet is modified to use this IP address as the source IP.

```
eqcli> vlan vlan-name subnet subnet-name nat from ip_cidr out 1.2.3.33 nat
subnet-name out gw 10.0.0.254
```

Outbound NAT means that now we are taking packets from the internal network and sending them out of the external network. This means that the packets are routed, and we need to enable permissions between the networks:

```
eqcli > vlan internal subnet net permit external:net

eqcli: 12000287: Operation successful
```

```
   eqcli > vlan external subnet net permit internal:net
   eqcli: 12000287: Operation successful
```

Note that the permissions need to be set on both sides - the internal network is configured to allow traffic from the external network, and the external network is configured to allow traffic from the internal network.

Now we can analyze the changes to the running configuration that we have made. First, we enabled Outbound NAT:

```
   IP NAT Rules:
```

```
   List of active MAP/Redirect filters:
   map wm0 192.168.211.0/24 -> 10.0.0.68/32 proxy port ftp ftp/tcp
   map wm0 192.168.211.0/24 -> 10.0.0.68/32 portmap tcp/udp auto
   map wm0 192.168.211.0/24 -> 10.0.0.68/32
```

All three rules are created for the single NAT change that we made. They can be read as "whenever traffic is leaving through the wm0 interface, if it has a 192.168.211.0 network source IP address, change the source IP address to 10.0.0.68".

Second, we changed the default gateway:

```
Source Routing Table:


 0.0.0.0/00:
     default via 10.0.0.254


192.168.211.0/24:
     default via 10.0.0.254


 10.0.0.0/24:


     default via 10.0.0.254
```

Both networks now use the same default gateway, since all traffic will be sent through that router.

Third, we added permit rules for the networks:

```
 IPv4 Rules:

 1: pass on interface lo0 all hits: 0 bytes: 0


 2: pass on interface wm1 hits: 90 bytes: 4156
                   From To
      192.168.211.0/24 192.168.211.0/24
                           -> 10.0.0.0/24
                                  0.0.0.0/0


 3: pass on interface wm0 hits: 6 bytes: 295
                   From To
          10.0.0.0/24 10.0.0.0/24
```

```
            0.0.0.0/0 -> 0.0.0.0/0
                                   192.168.211.0/24
 4: block on interface wm1 hits: 0 bytes: 0
                  From To
      192.168.211.0/24 192.168.211.0/24
                          -> 10.0.0.0/24
                                 0.0.0.0/0
 5: pass on interface wm1 hits: 0 bytes: 0
                  From To
      192.168.211.0/24 -> any

 6: block on interface wm0 hits: 0 bytes: 0
                  From To
           10.0.0.0/24 192.168.211.0/24
                          -> 10.0.0.0/24
                                0.0.0.0/0
 7: pass on interface wm0 hits: 3 bytes: 517
                  From To
           10.0.0.0/24 -> any
 8: pass on interface wm0 hits: 0 bytes: 0
                  From To
                   any 192.168.211.0/24
                          -> 10.0.0.0/24
                                0.0.0.0/0
9: block all hits: 0 bytes: 0
```

The main difference between these rules and those in Dual VLAN/Network with 2

Gateways is that because of the new permissions, Rules 2 and 3 now include both networks in them, meaning that traffic can be sent to either network rather than just one. Additionally, rule 8 has replaced two separate rules, because all traffic coming from the Internet will now enter Equalizer through the wm0 interface.

This configuration corresponds to the same scenario as Standard Dual Network configuration, but with the requirement that the internal servers are required to be able to access the Internet.

Using VLANs

Many networking technologies use a technique called broadcasting to provide services on a Local Area Network (LAN). Like traditional television or radio signals that are broadcast over the airwaves, broadcast network transmissions are received by every node on the same LAN segment, or broadcast domain. The Address Resolution Protocol (ARP), the Dynamic Host Configuration Protocol (DHCP), and the Router Information Protocol (RIP) are all examples of protocols that provide network services through broadcasting.

A LAN is a single broadcast domain composed of all the systems that are physically connected to the same switches, hubs, and other devices that communicate at the Data Link Layer (Layer 2) of the OSI Networking Model. These devices communicate using Layer 2 protocols, like Ethernet and ARP.

Virtual Local Area Network (VLAN) technology was developed to overcome these physical limitations of traditional LAN technology. A VLAN is essentially a means of grouping systems at the Data Link Layer (Layer 2 of the OSI networking model), using methods that are independent of the physical connection of the device to the network.

By exchanging broadcast packets -- packets that are essentially sent to all systems connected to a Layer 2 switching device -- switches can maintain a list of all MAC addresses connected to them and to the other switches to which they are connected. A set of Layer 2 devices and the systems connected to them form a broadcast domain -- meaning that all the systems can talk to one another using broadcast packets.

Conversely, broadcast packets are not forwarded beyond the boundaries of the broadcast domain. For example: if two LANs are connected by a router (a Network Layer, or Layer 3, device), the broadcast traffic for one LAN is never forwarded to the other LAN. The layout of a traditional LAN is therefore restricted to those systems that can be wired together using Layer 2 devices -- a physically distant system that requires connectivity to the LAN would require special routing and address translation (at Layer 3) in order to reach the LAN.

The dependence of LAN technology on physical connectivity at Layer 2 leads to two basic difficulties:

- Broadcasts are received by all systems in the broadcast domain - and if there is sufficient broadcast traffic, it can significantly reduce the overall performance of the LAN, to the point where some services may simply not be able to function properly due to latency or other factors introduced by a high level of broadcast traffic.
- If you want to include a system that is not physically connected to the LAN in the LAN’s broadcast domain, you need to physically connect the system to the LAN.

One problem with broadcasting is that lots of broadcast traffic on a LAN can slow network traffic down, as well as slow individual systems down. If there is so much broadcast traffic on the LAN that other non-broadcast traffic is significantly delayed (or never delivered), this is called a broadcast storm. Broadcast storms typically arise when network loops are created through faulty network configuration, but can also happen as the result of a malicious attack. For example, a classic Denial of Service attack is to send an ICMP echo request ("ping") over the LAN that specifies the source address of a system and a broadcast address for the destination. Every system receiving the ping will respond to it -- flooding the system specified as the source of the ping with ICMP echo replies.

There are also other security concerns associated with broadcasting. Since all the systems in the broadcast domain can see broadcast packets, the information in them is susceptible to discovery, intercept, and modification. This is of particular concern in industrial Ethernet environments (where, for example, manufacturing processes are controlled directly by computers) and in any environment (such as government and finance) where sensitive data is regularly transmitted over the LAN.

A number of methods can be used to mitigate problems and threats associated with large broadcast domains, including broadcast filtering and physically separating large broadcast domains into smaller domains. The problem with these solutions is that the are typically implemented at the Network Layer (Layer 3), and require Layer 3 devices (such as routers and firewalls) to implement them. These Layer 3 devices require separate subnets, and themselves emit a significant amount of broadcast traffic.

What we really want is a way of abstracting the idea of a LAN so that large broadcast domains can be separated into smaller domains *without requiring any network rewiring or physical movement of systems*. We’d also like the ability to extend broadcast domains across Layer 3 devices to physically remote systems.

With a VLAN, the broadcast domain for a particular system is determined by the software settings on the Layer 2 switch port to which the system is connected.

So, for example, in a traditional LAN, all the systems connected to Switch A would be part of Broadcast Domain A. If the switch is a VLAN-capable switch, then it is possible to configure several ports on the switch for VLAN A, several others to VLAN B, others to VLAN C, and so on.

This allows you to both:

- reduce the number of devices in local broadcast domains
- extend broadcast domains across devices separated by more than one switch

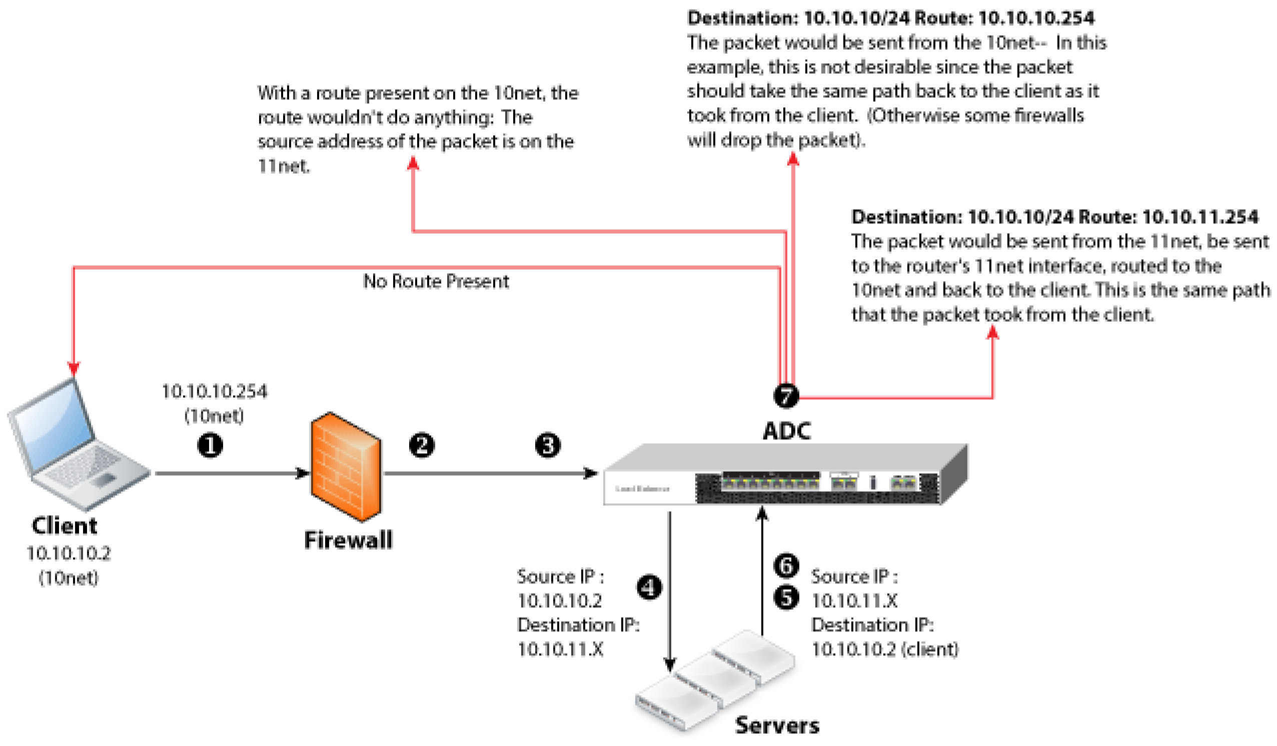
The predominant VLAN standard is 802.1q. This standard adds a VLAN tag to the information in the Ethernet packet. Since they operate at the switching level, VLANs are Layer 2 technologies -- though they are often confused with Layer 3 subnetting, because in many configurations there is one VLAN configured per subnet. This is usually done for the practical purpose of allowing the systems on a VLAN to be managed as a group by other network management devices/software that work by IP address ranges, for example, rather than VLAN tags.

How the ADC Routes a Packet

When an ADC sends out a packet, it determines how to send it as follows:

1. The ADC determines whether the packet destined for a system is directly attached to one of the configured networks.
    a. If "Yes", then it needs to determine whether there is a preferred route that specifies how to send it.
        i. If "Yes", it uses that route.This means that the ADC will send the packet to the router-- using whatever method it has of communicating with the router. If you use a router that's on one network, it will send out that packet through the interface connected to that network.. If you use a router on another configured network with another IP address, it will send it out of the interface attached to that address.
        ii. If "No", it sends directly (ARP for the address and then send to the MAC address via Ethernet)
    b. If "No," it searches the routes present for the source network that this packet has in "most-specific" to "least-specific" order. It determines whether a route exists that matches both the packet's source network and the destination network.
        i. If "Yes," it uses that route.
        ii. If "No", it drops the packet and returns an error.

Consider the following example using the illustration below. In the example, two subnets are used. The "10net" (IP 10.10.10./24) and the "11net" (IP 10.10.11/24).


Destination: 10.10.10/24 Route: 10.10.10.254
The packet would be sent from the 10net-- In this example, this is not desirable since the packet should take the same path back to the client as it took from the client. (Otherwise some firewalls will drop the packet).
With a route present on the 10net, the route wouldn't do anything: The source address of the packet is on the 11net.
Destination: 10.10.10/24 Route: 10.10.11.254
The packet would be sent from the 11net, be sent to the router's 11net interface, routed to the 10net and back to the client. This is the same path that the packet took from the client.
No Route Present
10.10.10.254
(10net)
❶
❷
❸
❼
ADC
Client
10.10.10.2
(10net)
Firewall
Source IP :
10.10.10.2
Destination IP:
10.10.11.X
❹
❻
❺
Source IP :
10.10.11.X
Destination IP:
10.10.10.2 (client)
Servers

❶ The client with IP address 10.10.10.2, sends a packet to a cluster with IP address 10.10.11.21, through a firewall with IP address 10.10.10.254.

❷ The firewall forwards the packet out of it's 10.10.11.254 interface.

❸ The ADC receives the request through the cluster IP 10.10.11.21.

❹ The ADC forwards the request to the server (spoofed): with source IP address 10.10.10.2 and destination IP address 10.10.11.X.

❺ The server responds with a source IP address 10.10.11.X and a destination IP address 10.10.10.2 (the client).

❻ The response arrives at the ADC. It doesn't matter which interface it enters ; just the IP addresses in step 5.

❼ The ADC then needs to send the packet out:

a. With no route present, it will send it direct to 10.10.10.2 since it's attached to the 10net.
b. With a route present on the 10net, the route wouldn't be used because the source address of the packet is on the 11net.
c. With a route present on the 11net with:

   Destination: 10.10.10/24
   Route: 10.10.10.254

   The packet would be sent from the 10net--In this example, this is not desirable since the packet should take the same path back to the client as it took from the client. (Otherwise some firewalls will drop the packet).
d. With a route present on the 11net that looks like this:

   Destination: 10.10.10/24
   Route: 10.10.11.254

   The packet would be sent from the 11net, be sent to the firewall's 11net interface, routed to the 10net and back to the client. This is the same path that the packet took from the client.

Configuring Front Panel Ports

Front panel ports are configured using either the CLI or GUI.

By default, all switch ports are configured as follows:

- full duplex
- full autonegotiation (Equalizer will attempt to auto negotiate the highest available speed with the unit on the other end of the connection)

If needed, ports can be configured to match specific port settings required by the server connection. For example, you could use the switch interface to configure a particular switch port to be 100Mb/s and half-duplex to accommodate older hardware.

Supported 10Gb Media Subtypes

10Gb ports are available on Equalizer E670LX and E970LX. The following media subtypes are supported:

10GbaseLR - single-mode fiber
10GBase-SR 850nm Multi-mode
10GBase CX4 copper
10GBase Twinax copper
10GBase Twinax Long copper
10GBase-LRM 850nm Multi-mode
10GBase-T - RJ45

Viewing Link Status and Port Settings

Refer to "Interface Commands" on page 195 for a complete listing of the CLI Interface commands.

Viewing Link Status and Port Settings(CLI)

The current link status of each port as well as the current settings, use the "show interface" command as in this example below:

```
eqcli > show interface
Interface  Duplex Mode  Speed  Status
ge01       NA           NA     Link Down
ge02       NA           NA     Link Down
ge03       NA           NA     Link Down
ge04       NA           NA     Link Down
ge05       NA           NA     Link Down
ge06       NA           NA     Link Down
ge07       NA           NA     Link Down
ge08       NA           NA     Link Down
xe01       full         10G    Link Up
xe02       full         10G    Link Up
mgmt       full         100M   Link Up
eqcli >
```

The same information for a single port can be displayed by specifying the port name:

```
eqcli > show interface ge01
Interface Number         : ge01
Duplex mode              : NA
Link Speed               : NA
Actual Link Status       : Link Down
Configured Link Status   : Link Up
Maximum MTU              : 9000
Maximum Speed            : 1G
eqcli >
```

Port settings are as follows:

- **Duplex Mode** - If the port status is Link Up, this is the current port duplex setting. If the status is Link Down, this is either the highest duplex that can be negotiated, or the force setting. Can be set to full or half.
- **Link Speed** - If the port status is **LinkUp**, this is the current port speed. If the status is **Link Down**, this is the highest speed that can be negotiated, or the **Force** setting. Can be set to **10**, **100**, or **1000** Mbits.

Viewing Link Status and Port Settings(GUI)

Refer to "Interfaces" on page 296 for details on using the GUI for this function.

Viewing Link Status and Port Settings (E350GX, E450GX, E650GX Only)

Viewing Link Status and Port Settings(CLI)

The current link status of each port as well as the current settings, use the "show interface" command as in this example below:

```
eqcli > show interface
Interface  Autonegotiation Mode  Duplex Mode  Speed  Status
if01       full                  full         1G     Link Up
if02       NA                    NA           NA     Link Down
if03       NA                    NA           NA     Link Down
if04       NA                    NA           NA     Link Down
if05       NA                    NA           NA     Link Down
if06       NA                    NA           NA     Link Down
if07       NA                    NA           NA     Link Down
if08       NA                    NA           NA     Link Down
if09       NA                    NA           NA     Link Down
if10       NA                    NA           NA     Link Down
if11       NA                    NA           NA     Link Down
if12       NA                    NA           NA     Link Down
```

The same information for a single port can be displayed by specifying the port name:

```
eqcli > show interface if01
Interface Number         : if01
Autonegotiation mode     : full
Duplex mode              : full
Link Speed               : 1G
Actual Link Status       : Link Up
Configured Link Status   : Link Up
Maximum MTU              : 1500
Maximum Speed            : 1G
eqcli >
```

Port settings are as follows:

- Autonegotiation mode - Use one of the following:

  full - Full autonegotiation at all supported speed and duplex settings.

  select - Autonegotiation at the current speed and duplex parameter settings only.

  force - Set the port to the current speed and duplex parameter settings with no autonegotiation.

  Whether you choose **full**, **select**, or **force** depends on the operating characteristics of the device on the other end of the connection. Check the documentation for the other device and try to match the settings as much as possible on both sides of the connection.

- **Duplex Mode** - If the port status is Link Up, this is the current port duplex setting. If the status is Link Down, this is either the highest duplex that can be negotiated, or the force setting. Can be set to full or half.
- **Link Speed** - If the port status is **Link Up**, this is the current port speed. If the status is **Link Down**, this is the highest speed that can be negotiated, or the **Force** setting. Can be set to **10**, **100**, or **1000** Mbits.

Displaying Port Statistics

Displaying Port Statistics (CLI)

Use the interface context stats command to display statistics for a particular port, as in this example:

```
eqcli > interface if01 stats
Transmitted Counters:
packets           : 314966
bytes             : 422
multicasts        : 2
errors            : 0
collisions        : 0
Received Counters:
packets           : 3669409
bytes             : 266
multicasts        : 759068
errors            : 0
drops             : 0
unknown protocol  : 0
eqcli >
```

Displaying Port Statistics (GUI)

Refer to "Interfaces" on page 296 for details on using the GUI for this function.

Source Based Routing Scenarios

Source routing allows the originator of a packet to partially or completely specify the path that a packet will take through a network, as well as the return path. In contrast, non-source-routing devices determine that path based on the packet’s destination. Source routing allows:

- Easier troubleshooting
- Improved traceroute
- Enables a node to discover all the possible routes to a host.
- Allows a source to directly manage network performance by forcing packets to travel over one path to prevent congestion on another.

Source routing requires careful management by the administrator when building the source address selection and source routing tables to ensure a coherent overall routing strategy. For this reason, it is often called policy routing, since routing behavior is determined by a collection of routing tables built by the administrator.

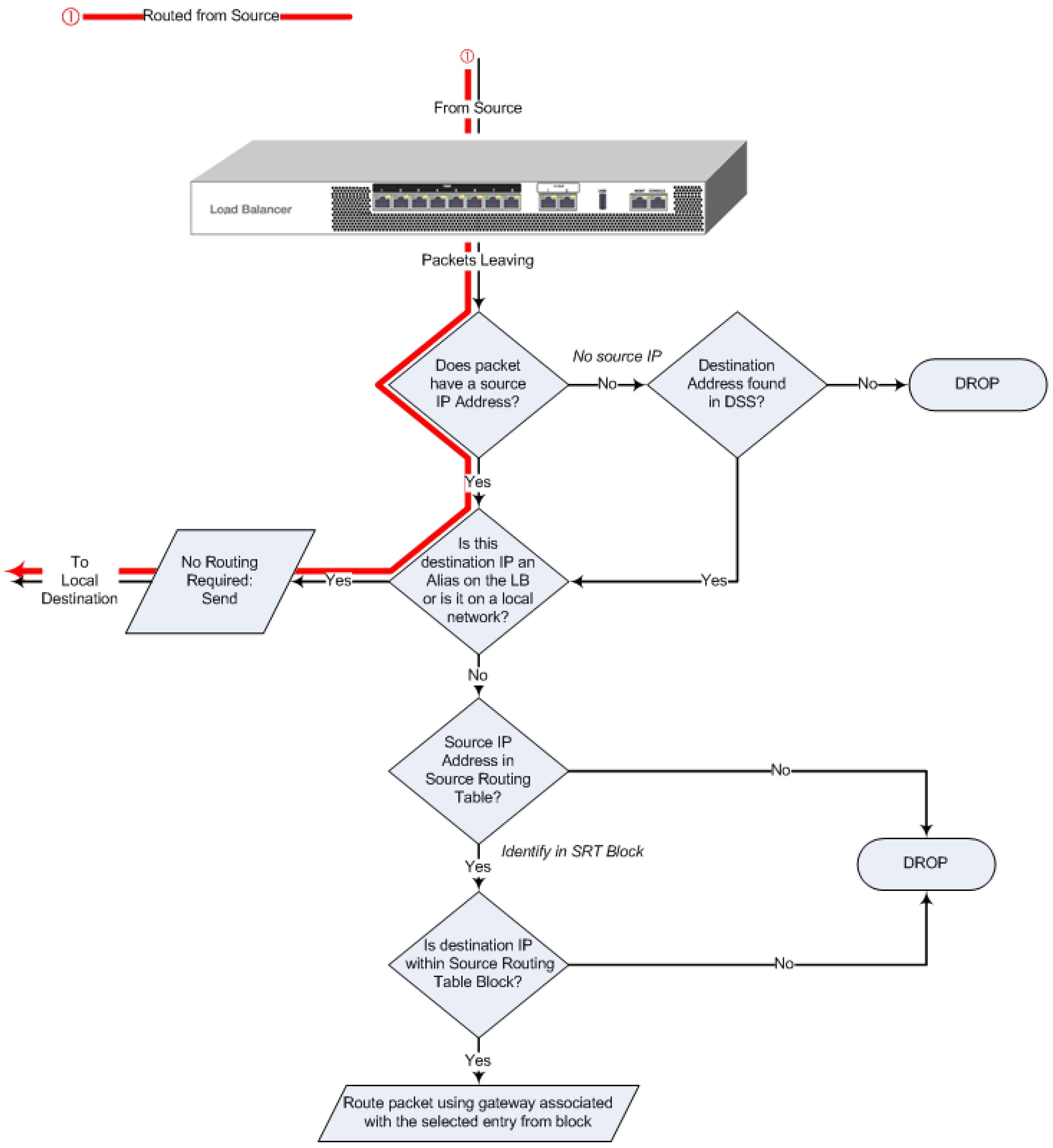
Source Selection

As a load balancing device Equalizer may change the source address in a packet, the destination address in a packet, or both, before sending a packet on to the next-hop gateway. In doing so, it will perform source address selection to determine the appropriate source address to use when a packet is sent out on the network. (Refer to the illustration on the next page.)

Equalizer is frequently required to choose from a number of possible source addresses when sending packets. An example is when it sends a probe to a server behind it. Some servers will be on a network that is local to Equalizer, and so it will chose its IP address on the appropriate VLAN to use as the source address in a probe packet. If the server is not located on a network that is local to Equalizer, then Equalizer will consult the source address selection table to choose a source address, and route the packet according to the information in the source routing table.

Refer to "Load Balancing & Networking" on page 87for a detailed discussion of VLANs and configuration with Equalizer.

The figure below shows the general flow of a packet through Equalizer, demonstrating the various check points and destination selection techniques that are used. In "Source Routing Scenarios" on page 120 practical scenarios are presented in “Road Map” style to demonstrate the routing selection used.



①
Routed from Source
①
From Source
Load Balancer
Packets Leaving
Does packet have a source IP Address?
No source IP
No
Destination Address found in DSS?
No
DROP
Yes
Is this destination IP an Alias on the LB or is it on a local network?
Yes
Yes
No Routing Required: Send
To Local Destination
No
Source IP Address in Source Routing Table?
No
DROP
Yes
Identify in SRT Block
Is destination IP within Source Routing Table Block?
No
Yes
Route packet using gateway associated with the selected entry from block

Source Routing Scenarios

The following are possible scenarios for load balancing source-based routing through Equalizer:

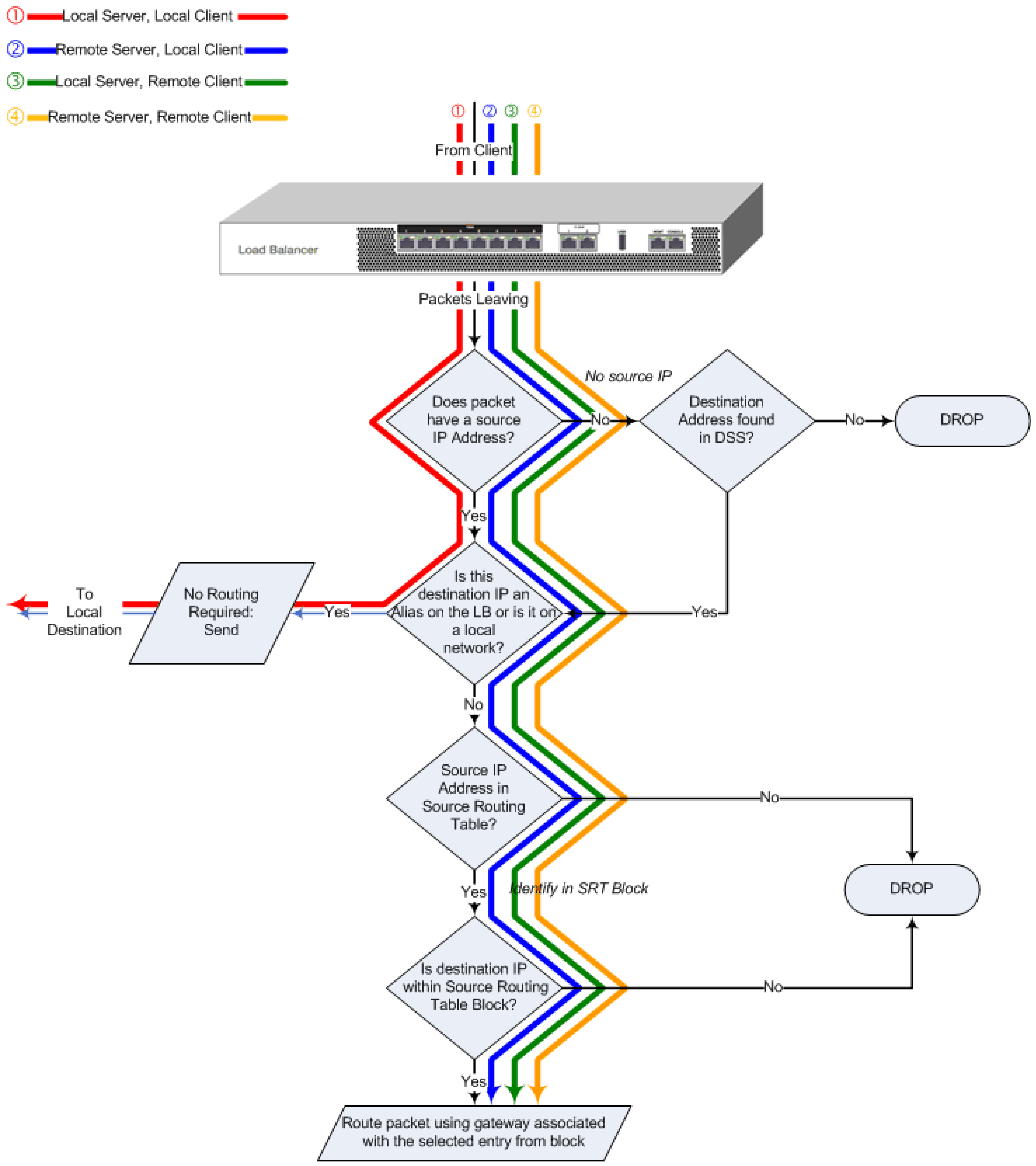
Scenario	Source	Destination	DSS Used
Spoof Load Balancing Toward Server 1. Local Server, Local Client 2. Routed Server, Local Client 3. Local Server, Remote Client 4. Remote Server, Remote Client	Client	Server	No
Non-Spoof Load Balancing Toward Server	Equalizer	Server	Yes
Spoof Load balancing Toward Client 1. Local Destination 2. Remote Destination	Cluster	Client	No
Non-Spoof Load Balancing Toward Client	Cluster	Client	No
Source, Destination Specified 1. Equalizer as Router	Source	Destination	No
Generated by Equalizer 1.IP Generated by Equalizer	Equalizer	Destination	Yes

Spoof Load Balancing Toward Server

In the load balancing source-based routing scenario presented below, spoofing is enabled so that the source is specified by a client IP and the destination is a server IP. (Refer to the illustration on the next page)

As indicated in the table above, four scenarios are possible:

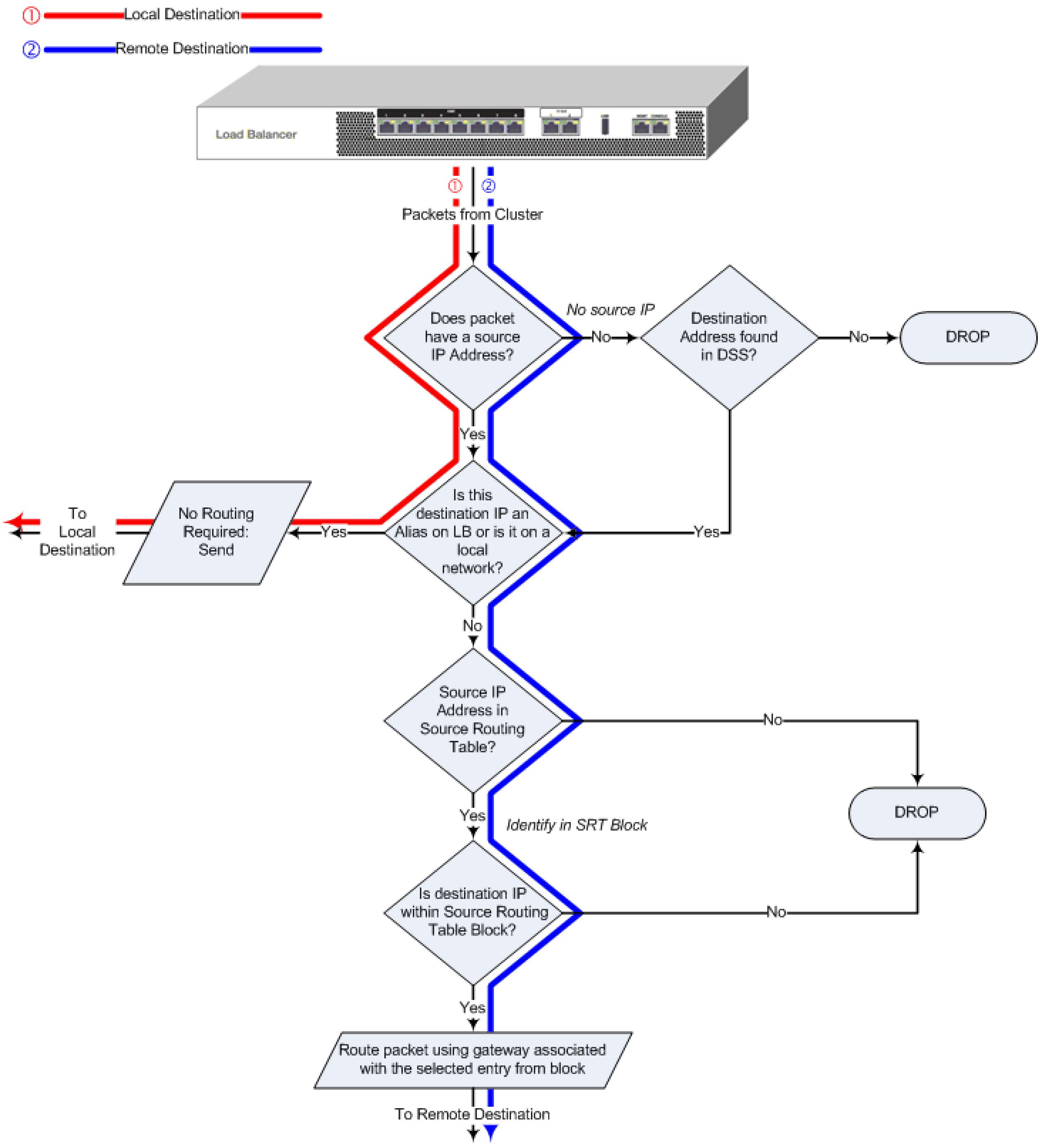
1. Local Server, Local Client - In this case the server is local and the client is local so no routing will be required. Since the packet has a source address and destination address has either an alias on Equalizer or is on the local network, the packets will be sent to the destination without routing.
2. Remote Server, Local Client - In this case the server is outside of the local network with a client that is local. The packet has a local source IP address. The server is not on the local network and therefore needs to be evaluated by the routing table to determine if the destination IP address is within the source routing table block. If it does lie within the block it will have a specific routing gateway associated with that block and will be routed using that gateway. If the destination does not lie within the source routing table block it will be dropped.
3. Local Server, Remote Client - In this case a server IP address lies within the local network while the client IP address is not within the local network.
4. Remote Server, Remote Client - In this case both the server and the client lie outside of the local VLAN.


① Local Server, Local Client
② Remote Server, Local Client
③ Local Server, Remote Client
④ Remote Server, Remote Client
① ② ③ ④
From Client
Load Balancer
Packets Leaving
Does packet have a source IP Address?
No
No source IP
Destination Address found in DSS?
No
DROP
Yes
Is this destination IP an Alias on the LB or is it on a local network?
Yes
No Routing Required: Send
To Local Destination
Yes
No
Source IP Address in Source Routing Table?
No
DROP
Yes
Identify in SRT Block
Is destination IP within Source Routing Table Block?
No
Yes
Route packet using gateway associated with the selected entry from block

Spoof Load Balancing Toward Client

In the load balancing source-based routing scenario presented below, spoofing is enabled so that the source is specified by a cluster and the destination is a client. (Refer to the illustration on the next page) Two scenarios are possible:

1. Local Destination- in this case the packets originating from a cluster and destined for a client has both a source IP address and a destination IP address that is on a local subnet. No routing is required for the packet. It is simply sent to the local address on the subnet.
2. Remote Destination- in this case a packet originating from a cluster and destined for a remote client does have a local source IP address yet the destination is not on a local subnet. The packet will be evaluated to see whether the source address/destination pairing is identified in a source routing table block. if it is not in the routing table the packet will be dropped. If the destination IP is identified in the source routing table then the packet will be sent using the gateway associated with the entry in the routing table.



① Local Destination
② Remote Destination
Load Balancer
①
②
Packets from Cluster
Does packet have a source IP Address?
No source IP
No
Destination Address found in DSS?
No
DROP
Yes
Is this destination IP an Alias on LB or is it on a local network?
Yes
Yes
No Routing Required: Send
To Local Destination
No
Source IP Address in Source Routing Table?
No
DROP
Yes
Identify in SRT Block
Is destination IP within Source Routing Table Block?
No
Yes
Route packet using gateway associated with the selected entry from block
To Remote Destination

Non-Spoof Load Balancing Toward Client

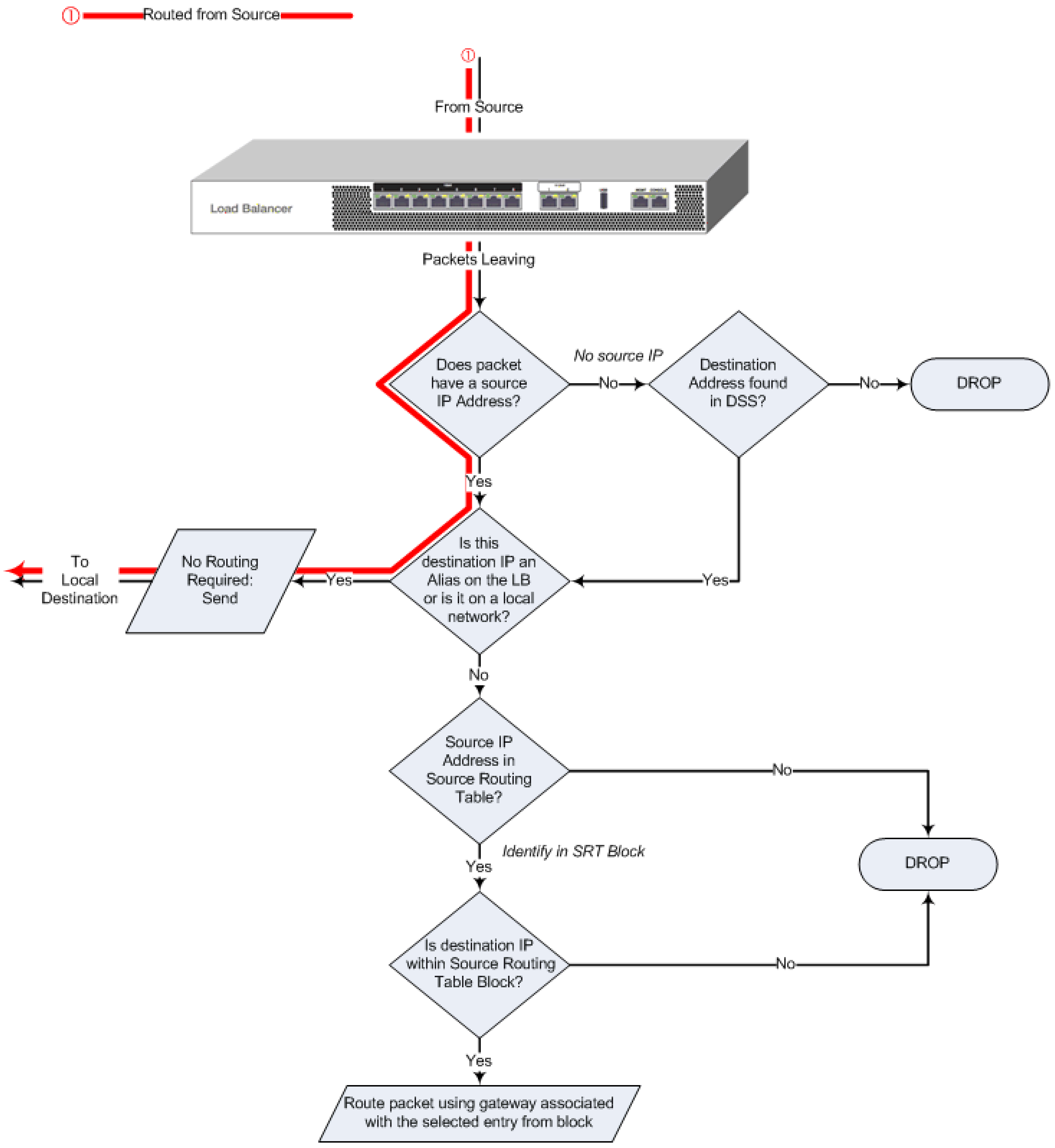
This scenario is the same as "Spoof Load Balancing Toward Client" on page 123 however, spoofing is disabled and the source is a cluster IP address and the destination is the load balancer's IP . The routing possibilities are the same as "Spoof" load balancing except that the remote clients are not applicable as the IP address is guaranteed to be "local".

Non Spoof Load Balancing Toward Server

This scenario is the same as "Spoof Load Balancing Toward Server" on page 121 except that in this scenario the source IP address is the load balancer's IP address. The routing possibilities are the same as "Spoof" load balancing except that the remote clients are not applicable as the IP address is guaranteed to be "local".

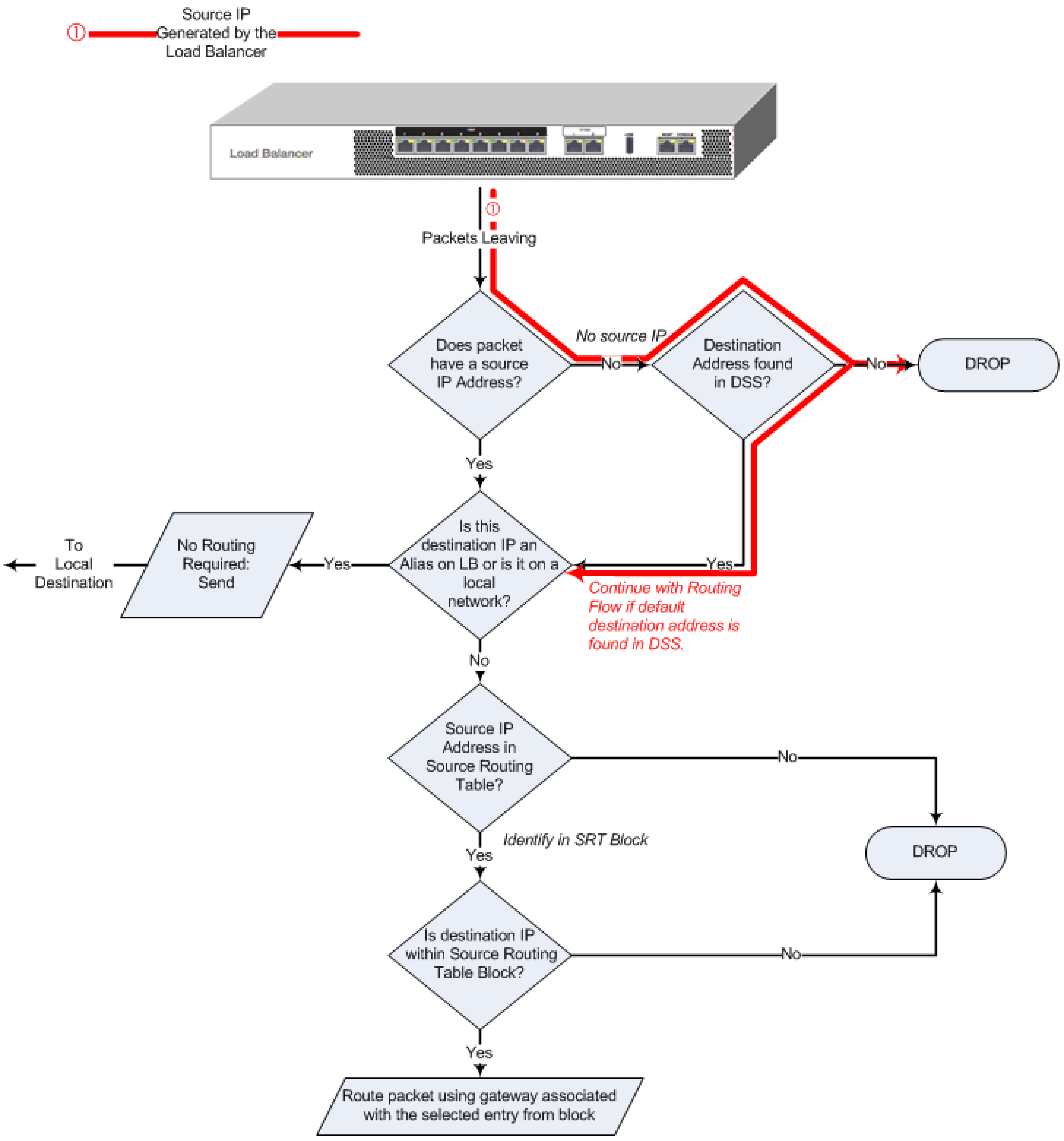
Source, Destination Specified

In this scenario, the source and destination are both specified by the client. Equalizer will function as a router to send the packet directly to the addresses specified. (Refer to the illustration on the next page.)


① Routed from Source
①
From Source
Load Balancer
Packets Leaving
Does packet have a source IP Address?
No source IP
No
Destination Address found in DSS?
No
DROP
Yes
Yes
Is this destination IP an Alias on the LB or is it on a local network?
Yes
No Routing Required: Send
To Local Destination
No
Source IP Address in Source Routing Table?
No
DROP
Identify in SRT Block
Yes
Is destination IP within Source Routing Table Block?
No
Yes
Route packet using gateway associated with the selected entry from block

Generated by Equalizer

This scenario is typically used for administrative and probing purposes. It can also be used for upgrades, pinging and Equalizer image updates. As shown below, a packet will be dropped if no source IP address is found. As shown below, the packet routing will be determined by the default gateway specified in the DSS table. (Refer to the illustration on the next page.)


①
Source IP Generated by the Load Balancer
Load Balancer
①
Packets Leaving
Does packet have a source IP Address?
No source IP
No
Destination Address found in DSS?
No
DROP
Yes
Is this destination IP an Alias on LB or is it on a local network?
Yes
No Routing Required: Send
To Local Destination
Yes
Continue with Routing Flow if default destination address is found in DSS.
No
Source IP Address in Source Routing Table?
No
DROP
Identify in SRT Block
Yes
Is destination IP within Source Routing Table Block?
No
Yes
Route packet using gateway associated with the selected entry from block

Enabling DNS

To enable the Domain Name Service (DNS), add a name server to the configuration. Name servers are added to the **name-server** list one at a time, with a maximum of three name servers in the list (Primary, Secondary, Tertiary). The following table shows you how to perform DNS tasks using the CLI and the GUI:

Task	Command / Procedure	
Add a DNS server	CLI	[1][2]`eqcli > `**`name-server [primary|secondary|tertiary]ip-address`**
	GUI	See "Parameters" on page 275
Remove an DNS server	CLI	`eqcli > `**`no name-server ip-address`**
	GUI	See "Parameters" on page 275
Remove all DNS servers	CLI	`eqcli > `**`no name-server`**
	GUI	See "Parameters" on page 275
Disable DNS	CLI	`eqcli > `**`no name-server`**
	GUI	See "Parameters" on page 275
Display DNS servers	CLI	`eqcli > `**`show`**
	GUI	See "Parameters" on page 275

[1] The IP addresses of the primary, secondary, and tertiary name-servers can be added on the same line in CLI syntax. For example:

```
eqcli >name-server primary ip secondary ip tertiary ip
```

[2]Note that removing all the servers from the name server list disables DNS.

Configuring NTP

Network Time Protocol, or NTP is a protocol designed to synchronize the clocks of computers over a network. NTP on Equalizer is compatible with servers running versions 1, 2, 3, or 4 of the NTP protocol. An RFC for NTPv4 has not been written; NTPv3 is described in RFC 1305.

On Equalizer, NTP is used primarily to time various operations, to ensure accurate timestamps on log entries (with respect to server and client log timing), and to allow for examination of the timing of log entries on two Equalizers in a failover configuration.

NTP on Equalizer works by polling an NTP server defined through the GUI. The time between polls of the NTP server is controlled by the **minpoll** and **maxpoll** NTP parameters, which default to 64 seconds (1 min 4 sec) and 1024 seconds (~17 mins), respectively. The behavior of NTP is to poll with a frequency starting at **minpoll** and then decrease polling frequency over time to **maxpoll**, as the accuracy of the local clock approaches the accuracy of the remote server clock. The time it takes for the polling delay to increase from **minpoll** to **maxpoll** will vary based on a number of factors, including the accuracy of the clocks on the client and server, network latency, and other timing factors.

NTP calculates when the local and remote system clocks are sufficiently in sync to begin increasing the polling delay towards **maxpoll**. When the accuracy of the two clocks is significantly different, or there is significant latency, for example, the two clocks may never be in sufficient agreement to increase the delay towards **maxpoll**. In this case, Equalizer will continue to sync approximately every 64 seconds. This behavior indicates that a different NTP server should be chosen.

NTP packets are very small and should not cause any problems with Equalizer or network operation, except as described in the following section on NTP and plotting.

NTP and Plotting

When you initially configure NTP, this may effectively disable plotting until NTP completes the initial synchronization of Equalizer’s system clock with the NTP server -- which may take from several hours to several days. This is because plotting depends on accurate timestamps in the plot log. Since initially NTP is adjusting the time at frequent intervals, the timestamps in the plot log may become out of sync with the system clock, and so no plot data may be returned. Once NTP is no longer making adjustments to the system clock, plotting will function normally.

Default NTP Configuration

Equalizer is delivered with a default Network Time Protocol (NTP) configuration: the NTP daemon (**ntpd**) is enabled by default and the NTP server is set to **pool.ntp.org**. However, NTP will not be able to synchronize time with an NTP server until DNS is configured and working.

Rather than point at a single NTP server, most organizations use an NTP server pool defined by the NTP Pool Project.

Selecting an NTP Server

We recommend that you specify NTP pool servers appropriate for your geographic location. Selecting a pool server means that you are specifying an alias that is assigned by the NTP Pool Project to a list of time servers for a region. Thus, NTP pool servers are specified by geography. The following table shows the naming convention for servers specified by continent:

Worldwide - pool.ntp.org

Asia - asia.pool.ntp.org

Europe - europe.pool.ntp.org

North America - north-america.pool.ntp.org

Oceania - oceania.pool.ntp.org

South America - south-america.pool.ntp.org

To use the continent-based NTP pool servers for Europe, for example, you could specify the following pool servers in Equalizer's time **Configuration** screen (tab):

```
0.europe.pool.ntp.org
1.europe.pool.ntp.org
2.europe.pool.ntp.org
```

You can also specify servers by country. So, for example, to specify a UK based time server pool, you would use:

```
0.uk.pool.ntp.org
1.uk.pool.ntp.org
2.uk.pool.ntp.org
```

Or, for the US, you would use:

```
0.us.pool.ntp.org
1.us.pool.ntp.org
2.us.pool.ntp.org
```

Be careful when using country based NTP pool servers, since some countries contain a very limited number of time servers. In these cases, it is best to use a mix of country and continent based pool servers. If a country has only one time server, then it is recommended you use a time server pool based in another nearby country that supports more servers, or use the continent based server pools.

For example, Japan has 6 (six) time servers as of the date this document was published. The organization that maintains time server pools recommends using the following to specify time server pools for Japanese locations:

```
2.jp.pool.ntp.org
0.asia.pool.ntp.org
2.asia.pool.ntp.org
```

For more information on choosing NTP pool servers, please see the NTP pool server web pages at:

http://support.ntp.org/bin/view/Servers/NTPPoolServers

For general information on the NTP Pool Project, please go to the project home page:

http://www.pool.ntp.org/

Managing NTP

The following table shows you how to perform NTP tasks using the CLI and the GUI:

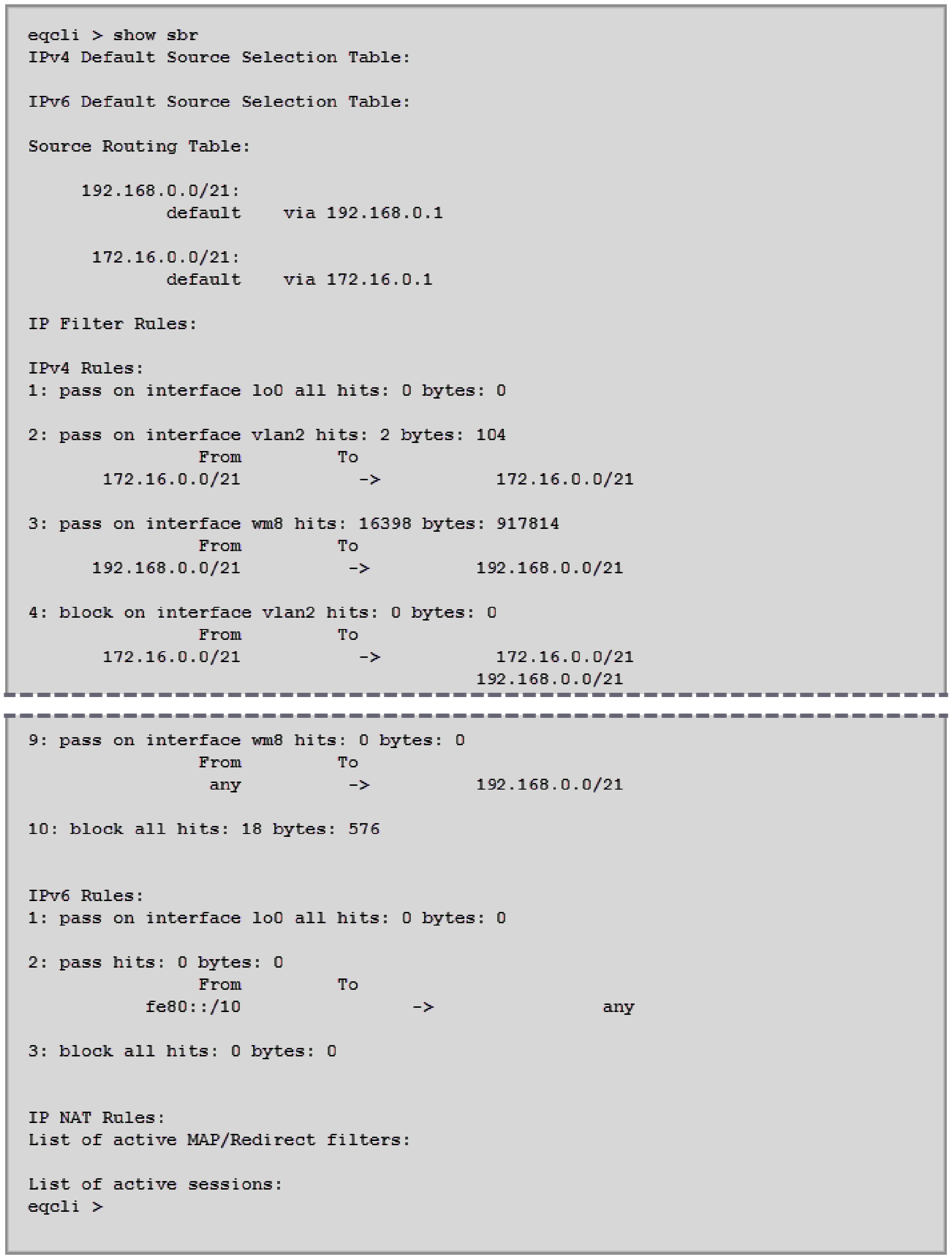
Task	Command / Procedure	
Add an NTP server	**CLI**	`eqcli > ntp-server` *`name`* The *name* parameter can be an NTP server name or an NTP pool name.
	GUI	Click **Hostname> Maintenance > NTP**. Enter an **NTP Server** or pool name. Click **Commit**.
Remove the NTP server	**CLI**	`eqcli >` **`no ntp-server`**
	GUI	Not implemented.
Disable NTP	**CLI**	`eqcli >` **`ntp disable`**
	GUI	Not implemented.
Enable NTP	**CLI**	`eqcli >` **`ntp enable`**
	GUI	Not implemented.
Display NTP server	**CLI**	`eqcli >` **`show`**
	GUI	Not implemented.

Source Routing Tables & Rules

A "Source Routing Table" on page 135 is a table that identifies how a packet should be sent by the system based on incoming route information.

Rules in include "IP Filter Rules" on page 136, which govern the IP traffic flow into and out of the system and includes IPv4 or IPv6 Rules, and "IP NAT Rules" on page 139, which are processed when a packet is exiting the system.

All of this information can be viewed on the same CLI output by entering the following:



```
eqcli > show sbr
IPv4 Default Source Selection Table:

IPv6 Default Source Selection Table:

Source Routing Table:

     192.168.0.0/21:
             default    via 192.168.0.1

      172.16.0.0/21:
             default    via 172.16.0.1

IP Filter Rules:

IPv4 Rules:
1: pass on interface lo0 all hits: 0 bytes: 0

2: pass on interface vlan2 hits: 2 bytes: 104
                From         To
       172.16.0.0/21           ->           172.16.0.0/21

3: pass on interface wm8 hits: 16398 bytes: 917814
                From         To
      192.168.0.0/21          ->          192.168.0.0/21

4: block on interface vlan2 hits: 0 bytes: 0
                From         To
       172.16.0.0/21           ->           172.16.0.0/21
                                          192.168.0.0/21
```

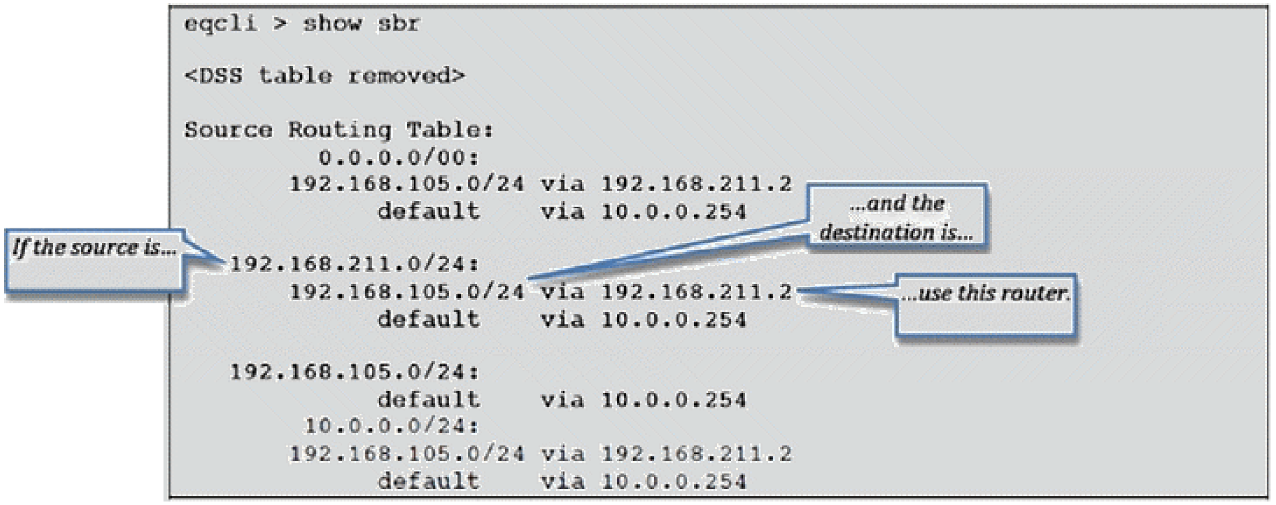
```
9: pass on interface wm8 hits: 0 bytes: 0
                From         To
                 any          ->          192.168.0.0/21

10: block all hits: 18 bytes: 576


IPv6 Rules:
1: pass on interface lo0 all hits: 0 bytes: 0

2: pass hits: 0 bytes: 0
                From         To
           fe80::/10                ->                any

3: block all hits: 0 bytes: 0


IP NAT Rules:
List of active MAP/Redirect filters:

List of active sessions:
eqcli >
```

Source Routing Table

The *sroute* table, or Source Routing Table is an excellent tool for identifying how a packet should be sent by the system. It is an aggregation of routing or all subnets and destination networks that you configure.

The Source Routing table is displayed as part of the CLI output when using the **`show sbr`** command. An example is shown below:

Note - The example below is a truncated example of the **`show sbr`** command display showing the Source Routing Table display only.



```
eqcli > show sbr

<DSS table removed>

Source Routing Table:
        0.0.0.0/00:
      192.168.105.0/24 via 192.168.211.2
            default    via 10.0.0.254

   192.168.211.0/24:
      192.168.105.0/24 via 192.168.211.2
            default    via 10.0.0.254

   192.168.105.0/24:
            default    via 10.0.0.254
        10.0.0.0/24:
      192.168.105.0/24 via 192.168.211.2
            default    via 10.0.0.254
```

In the example above traffic that is sourced from all local networks is sent through the 10.0.0.254 gateway, unless it is destined for the 192.168.105.0/24 destination network. Because the default gateway for the 192.168.211.0/24 local network is on the 10.0.0/24 local network, there is an outbound NAT configuration between these two networks.

IP Filter Rules

The current IP Filter rules are displayed as part of the CLI output when using the **`show sbr`** command. An example is shown below. The example is shortened due to its length.

Note - The example below is a truncated example of the **`show sbr`** command display. In addition to the IP Filter Rules, Default Source Selection Table, the IPv6 Default Selection Table, IPv6 Rules, and IP NAT rules will also be displayed.

```
IP Filter Rules:
IPv4 Rules:
1: pass on interface lo0 all hits: 287 bytes: 14900

2: pass on interface wm1 hits: 11394 bytes: 326068
                From                    To
                192.168.211.0/24        192.168.211.0/24
                192.168.105.0/24 ->     192.168.105.0/24
                                        10.0.0.0/24
                                        0.0.0.0/0

3: pass on interface wm0 hits: 120406 bytes: 7689819
                From                    To
                10.0.0.0/24             10.0.0.0/24
                0.0.0.0/0       ->      0.0.0.0/0
                                        192.168.211.0/24
                                        192.168.105.0/24

4: block on interface wm1 hits: 0 bytes: 0
                From                    To
                192.168.211.0/24        192.168.211.0/24
                                ->      192.168.105.0/24
                                        10.0.0.0/24
                                        0.0.0.0/0
```

The example above shows each filter rule, along with the groups of networks that the rule applies to, and the number of times each rule has been used (and bytes that have been received using this rule).

Each column of From and To addresses can be viewed as an "or" group. For example, rule #3 can be read as:

“Allow traffic on interface wm0 which is from either the 10.0.0.0/24 network or the 0.0.0.0/0 network, and is destined for either the 10.0.0.0/24, the 0.0.0.0/0, the 192.168.211.0/24, or the 192.168.105.0/24 network.”

Rules are processed (and must be read) in order, from first to last. This means that as soon as a packet matches a particular rule it is used and Equalizer either passes or allows that packet, depending on the rule.

The individual rules are somewhat complicated and will be explained in "Load Balancing & Networking" on page 87

To summarize, rules are processed in numerical order by the packet filter. Pass rules cause packets to be allowed into the system and block rules are ones that explicitly block traffic from entering the system. The last rule is block in all which means that if a pass rule has not yet matched this particular packet, it will be dropped.

Using this command while trying to establish a connection that may not be working can be a good method of finding out what is wrong. In this example, 0 packets were blocked by the filter in rule 4 because rules 2 and 3 allowed all packets needed. If there is a misconfiguration, seeing packets being blocked can be a hint of what is wrong.

Enabling/Disabling IP Filter Rules

When you create a subnet, IP Filter (firewall) rules are automatically generated. An example is shown above. An option is available to disable these rules that may be used for troubleshooting or diagnostic purposes. Disabling the firewall turns off all system packet filtering . Any subnet permit/deny rules are ignored and all traffic will be routed between subnets.

- If you use the GUI, you will note that if you disable these rules and you navigate to **System > Network > VLANs > {any subnet} >Permitted Subnets** the following message will be displayed in red text at the top of the tab:

  "Firewall rules are currently disabled. Any 'Permit' and 'Deny' selections made below will be ignored until firewall rules are enabled."

- If you use the CLI, you will note that when you enter `eqcli >` **`show firewall`**,the state will be `Disabled`.

The rules are enabled by default.

To disable in the CLI, enter the following:

```
eqcli > firewall disable

eqcli: 12000287: Operation successful
```

To verify that the firewall (IPv4 Rules) have been disabled, enter the following:

```
eqcli > show firewall

Variable          Value
state             Disabled
```

To disable using the GUI:

1. Log in to the GUI.
2. In the left navigational pane, select **System > Global > Parameters.**
3. Click on **Global** to expand the branch and then select **Parameters** to display the **Parameters** configuration screen on the right.
4. Select the **Disable** option in the **Firewall Rules** group.

1. Click on **Commit** to save the change.

Verify that the rules have been disabled by checking the subnets, as shown above, and verify that the disabled message appears.

Note - This is not related to the IP Reputation (Refer to "IP Reputation" on page 259) filtering or any white/blacklists that may exist as part of that configuration.

IP NAT Rules

Equalizer performs outbound NAT by creating IP NAT rules. These rules are processed when a packet is exiting the system -unlike IP Filter rules which are processed when a packet is entering the system. When NAT is enabled, the system automatically generates NAT rules to support the specified configuration. The rule types are labeled `proxy port`, `ftp`, `ftp/tcp`, `tcp/udp`, etc.

These rules can are also displayed as part of the CLI output when using the **`show sbr`** command. An example is shown below:

Note - The example below is a truncated example of the **`show sbr`** command display. In addition to the IP NAT rules, Default Source Selection Table, the IPv6 Default Selection Table, IP Filter Rules, and IPv6 Rules will be displayed.

```
    IP NAT Rules:

    List of active MAP/Redirect filters:
    map wm0 192.168.211.0/24 -> 10.0.0.68/32 proxy port ftp ftp/tcp
    map wm0 192.168.211.0/24 -> 10.0.0.68/32 portmap tcp/udp auto
    map wm0 192.168.211.0/24 -> 10.0.0.68/32
    map wm0 192.168.105.0/24 -> 10.0.0.68/32 proxy port ftp ftp/tcp
    map wm0 192.168.105.0/24 -> 10.0.0.68/32 portmap tcp/udp auto
    map wm0 192.168.105.0/24 -> 10.0.0.68/32

    List of active sessions:
```

Three rules are added for each outbound NAT mapping. In this example, there are two mappings: one for the 192.168.211.0/24 local network and the other for the 192.168.105.0/24 destination network.

In this example, the rules specify that any packets that are leaving the system through the `wm0` interface with a source IP address on either the 192.168.211.0/24 or 192.168.105.0/24 network should instead be sent with a source IP address of 10.0.0.68.

If there are any NAT connections active, they will be displayed in the list of active sessions.

Network Troubleshooting Tools

There are several tools useful for troubleshooting networking configurations on Equalizer. To simplify troubleshooting, Equalizer includes a single **eqcli** command (**`show sbr`**) that displays the output of these tools.

There are other ways to view the same information in **eqcli**, however, the show sbr command displays the actual running state of the system, whereas commands such as show vlan [X] subnet [*Y]* show the configuration information and not necessarily the running data if there is a problem.

Chapter 12

Working in the CLI

Sections in this chapter include:

Starting the CLI

The Equalizer Command Line Interface, CLI, gives you complete administrative control over Equalizer and is one of the major new features in EQ/OS 10. The GUI is also available to view and modify the configuration, however, not all administrative options have been enabled in the GUI.

The CLI can be used over either a serial connection or an SSH connection.

Logging In to the CLI Over a Serial Connection

To start the Equalizer CLI over a serial connection:

1. Connect the supplied serial cable to Equalizer’s front panel serial port and to a properly configured terminal or terminal emulator, as described in "Basic Configuration" on page 50.
2. Press the <Enter> key to display the login prompt:

```
Equalizer -- EQ/OS 10
Username:
```

3. Log in using an Equalizer user name and password. If this is the first time you are logging in, use the default administrative user name and password as shown below:

```
Username: touch
Password: touch
Login successful.

EQ/OS 10

 Copyright 2013 Fortinet, Inc.
 Welcome to Equalizer!

eqcli >
```

See the section "Working in the CLI" on page 145 to begin familiarizing yourself with the CLI command environment.

Logging In to the CLI Over an SSH Connection

To start the Equalizer CLI over an SSH connection:

1. Ensure that SSH login is enabled for the VLAN and subnet over which you want to establish an SSH connection.
2. Use SSH client software to open a connection with Equalizer using the enabled VLAN IP address and port 22. Specify the login **eqadmin**, as shown in the example command line below:

```
$ssh eqadmin@172.16.0.200
```

3. Upon successful SSH login, Equalizer displays the **Username** prompt. Enter an Equalizer login, such as the default login, **touch**:

```
Username: touch
```

4. Enter the password for the user name specified in the previous step:

```
Password: touch
```

1. If the user name and password is correct, Equalizer responds with:

```
Login successful.

EQ/OS 10

Copyright 2015 Fortinet, Inc.

Welcome to Equalizer!

eqcli >
```

See "Working in the CLI" on page 145 to begin familiarizing yourself with the CLI command environment.

Exiting the CLI

You must exit the CLI from the global context prompt (`eqcli >`):

- Enter `exit` or `<ctrl-d>` to exit and commit any queued changes.
- Enter `quit` to exit and discard all queued changes.

If you are in a lower context, repeatedly enter one of the above commands, as appropriate, until you exit the CLI. Once you exit the CLI, the login prompt is displayed.

Working in the CLI

The Equalizer command line interface, or CLI, was developed to be an easy to use, intuitive, and flexible command line interface. It was patterned after CLIs used in other common networking equipment, so if you’ve used a CLI on another network device (such as a router), you should quickly feel comfortable using eqcli.

The CLI provides a number of features that are designed to make working at the command line easier and more effective, as described in this section.

CLI Contexts and Objects

The Equalizer CLI is a context oriented command interface. This means that the commands available at any time (and the objects they affect) depend on the current context. The current command context is always indicated in the CLI prompt. When you start the CLI, the command prompt looks like this:

```
eqcli >
```

This indicates that you are in the global context -- all commands available in the CLI for all objects can be executed from this context, and you can also set parameters for global services (such as NTP, DNS, etc.). You can also change to other contexts, whose scope is limited to a specific object. For example, you can enter the cluster specific context for a cluster named **cl01** by typing:

```
eqcli > cluster cl01
eqcli cl-cl01>
```

The prompt above indicates that the cluster specific context for **cl01** is the current context. In this context, the only commands available are those that affect cluster **cl01**.

Note that only the first 4 characters of an object name appear in the eqcli prompt. For example, if you have a cluster named **mycluster**, then you would enter the cluster specific context for this cluster by typing:

```
eqcli > cluster mycluster
eqcli cl-myc*>
```

The asterisk (`*`) in the prompt indicates that there are more than 4 characters in the cluster name. To display the complete object name in any context, use the **context** command:

```
eqcli cl-myc*> context
The current context is: ‘mycluster’
eqcli cl-myc*>
```

In each context, you can perform operations on the objects and parameters that exist in that context (e.g., create, delete, modify, display, set). When you change to another context, the eqcli prompt changes to include the suffix indicated in the chart above for each context. For example, when you change to the server context, the eqcli prompt changes from "`eqcli >`" to "`eqcli sv>`".

Within each context shown above, you can also type in the name of an object (existing or new) to enter an object specific context that will allow you to edit only that object's settings.

So, for example, you can start eqcli and type **server** to change to the server context -- as indicated by the prompt, which changes to "`eqcli sv>`". Now, you can use the **list** command to list all the existing servers. If you then type in the name of one of the existing servers while in the server context, you will enter the server specific context for that existing server -- the prompt changes to "`eqcli sv-server_name>`" to indicate that you are in the server specific context for the server with the name `server_name`. You could also do this directly from the global context by typing:

```
eqcli > show server
eqcli > sv-server_name show
```

Note that the eqcli prompt reserves only four characters for object names. So, for example, if you have a server named **sv02**, the entire server name will be displayed in the prompt, as shown in this example:

```
eqcli > server sv02
eqcli sv-sv02>
```

If the object name is longer than four characters, eqcli displays the first three characters and an asterisk (*) to show that the name is longer than four characters. For example, if you have a server named **Server2**, the prompt will look as follows when you change to the **Server2** specific context:

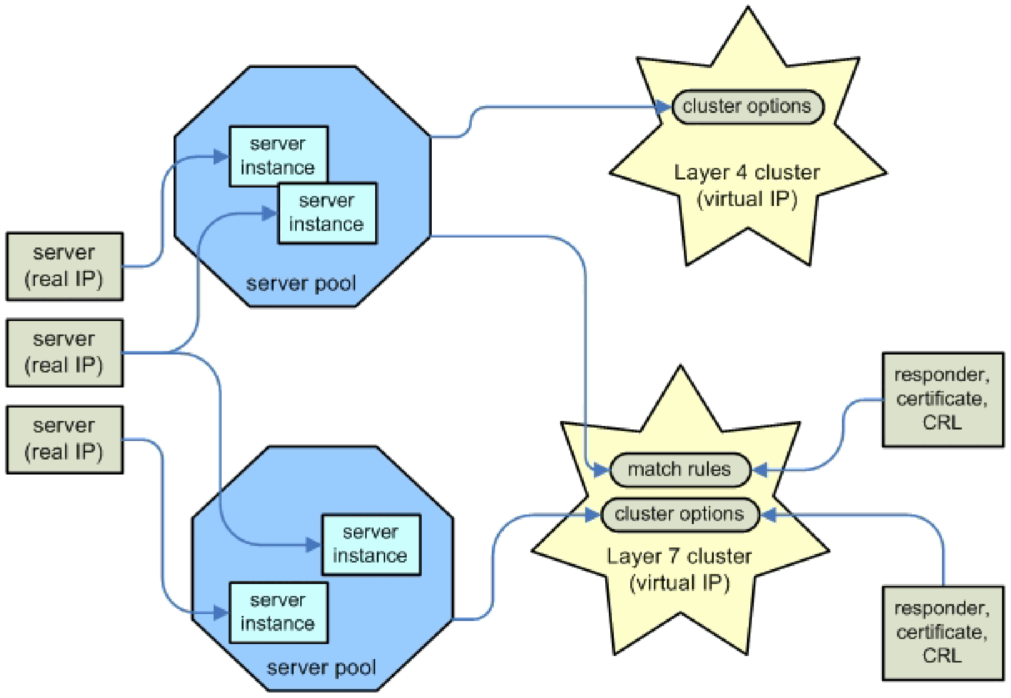
```
eqcli > server Server2
eqcli sv-Ser*>
```

The complete current context can always be displayed using the context command, as in this example:

```
eqcli > server Server2
eqcli sv-Ser*> context
The current context is: Server2
eqcli sv-Ser*>
```

Object Relationships

Most contexts in the CLI correspond to an Equalizer object -- **servers**, **server instances**, **server pools**, **clusters**, **match rules**, **responders**, **CRLs**, **certificates**. The following diagram shows the relationships among these objects.



On Equalizer, a **server** corresponds to a real server hosting an application behind Equalizer. Each server has an IP address that Equalizer uses to send client requests to the server. This IP address is sometimes called a "real IP" because it corresponds to a real server.

A server must be assigned to a **server pool** before it can be associated with a cluster. When you assign a server to a server pool, you create a **server instance** of that server in the server pool. The server instance definition specifies operating parameters for the real server that are effective only within that server pool. This allows you the flexibility to associate a single physical server with multiple server pools, and set different server instance options within each server pool.

A server pool in turn is assigned to a **cluster**. Client requests are sent to a cluster IP address (often called a "virtual IP") assigned to Equalizer and then routed to the server pool instance selected by the load balancing algorithm and other options. In all clusters, a server pool is assigned directly to the cluster. For Layer 7 clusters, additional alternate server pools, as well as other objects and options, can be assigned to one or more **match rules.**

A match rule is processed before cluster settings are processed, and behaves like an if-then statement: if a client request's content matches the conditional expression set in the match rule, then the options and objects specified in the match rule are used. If the expression in the match rule is not matched by the client request, then the next match rule is processed. If all match rules defined in the cluster are processed and none of them match the incoming request, then the objects and options set on the cluster are used to process the request.

The objects that can be selected by match rules include server pools, responders (used when no servers in a server pool are available), SSL certificates, and certificate revocation lists (CRLs). Many cluster options can also be specified in a match rule, including persistence settings and load balancing policy.

Supported operations on all objects are explained in "Context Command Summaries" on page 163.

Command Line Editing

Use the key sequences below to edit the current command line

ctrl-a **ctrl-e**	Move the cursor to the beginning of the line Move the cursor to the end of the line
ctrl-b **ctrl-f**	Move the cursor one character to the left Move the cursor one character to the right
esc-b **esc-f**	Move the cursor one word to the left (also left arrow) Move the cursor one word to the right (also right arrow)
ctrl-h **ctrl-k** **esc-d** **ctrl-u**	Delete the character to the left of the cursor Delete all characters from the cursor to the end of the line Delete the word to the right of the cursor Delete the entire line
ctrl-y	Inserts previously deleted text starting at the cursor
ctrl-t	Transpose the character under the cursor and the character to the left of the cursor
ctrl-l	Redraw the line
ctrl-n **ctrl-p**	Display next command from history (also up arrow) Display last command from history (also down arrow)

Entering Names for Equalizer Objects

Equalizer identifies administrative objects, such as clusters and servers, by name. The characters used in names are limited to standard ASCII letters ("A" through "Z" and "a" through "z"), numbers (0 through 9), and the characters "." (period), "-" (dash) and "_" (underscore), (*) asterisk, (@) "at" sign, and (/) backslash.

- The first character in a name must be a letter.
- Names can be at most 65 characters long.
- The readability of lists presented in the interface is increased by using short names that use as many unique characters at the beginning of the name as possible.

Using White Space in a Command Line

The CLI uses white space (i.e., one or more tab or space characters) as a delimiter between command line elements. To include spaces within a command line element (such as a string, a list of objects, or multiple flags), the entire element must be contained in double quotes. For example, this command line uses a space between the two server instances and two flags specified:

```
eqcli > srvpool sp01 si “sv01, sv02” flags “hot_spare, quiesce”
```

Enabling and Disabling Flags

Most objects have a flags keyword that is followed by one or more keywords that enable and disable particular object behavior. A single flag is specified as in this example:

```
eqcli> srvpool sp01 si sv01 flags hot_spare
```

Multiple flags in a command line can be separated using either a comma (**,**) or a vertical bar (**|**) between each flag. For example, all the following commands turn on the **hot_spare** and **quiesce** flags on a server instance:

```
eqcli> srvpool sp01 si sv01 flags hot_spare,quiesce
eqcli> srvpool sp01 si sv01 flags “hot_spare, quiesce”
eqcli> srvpool sp01 si sv01 flags hot_spare|quiesce
eqcli> srvpool sp01 si sv01 flags “hot_spare | quiesce”
```

Flags are disabled using the negate operator (the exclamation point character):

`!` Negates (turns off) the option that immediately follows it. No spaces are allowed between the negation operator and the option that follows.

For example, the following command disables the hot_spare option and quiesce options:

```
eqcli> srvpool sp01 si sv01 flags !hot_spare,!quiesce
```

A flag can be enabled and disabled in the object specific context or from any higher context. For example, you can type any of the following three command sequences to disable the **spoof** option on match rule **ma00** in cluster **cl00**:

```
eqcli > cluster cl00 match ma00
eqcli cl-cl00-match-ma00> flags !spoof# match rule specific context
eqcli > cluster cl00
eqcli cl-cl00> match ma00 flags !spoof# cluster context
eqcli > cluster cl00 match ma00 flags !spoof # global context
```

Command Abbreviation and Completion

You do not need to type an entire command name in order to execute a command. If you type enough characters to uniquely identify a command and then type a `<space>` or `<tab>` character, eqcli will automatically display the remainder of the command name.

For example, if in the global context you type `cert` and then press the space bar:

```
eqcli > cert<space>
```

The CLI fills in the rest of the command line for you, followed by a space:

```
eqcli > certificate<space>
```

This also works with multiple keywords on the same command line. So, for example, you can type the following:

```
eqcli > sh<space>cl<space>
```

And the CLI will expand this to:

```
eqcli > show<space>cluster<space>
```

If the string that you type before pressing `<space>` or `<tab>` does not uniquely identify a command, then the CLI displays a list of all the commands that match the string you entered, and then re-displays the string that you typed. For example:

```
eqcli > c<space>
certificate cluster context crl
eqcli > c
```

Detection of Invalid Commands and Arguments

Invalid commands and invalid arguments for specific commands are detected before they are committed and appropriate error messages are displayed.

Specifying Multiple Server Instances

When specifying server instances on the command line, the user can specify either a single object or a comma separated list of objects. For example, to create server instances of two servers (**sv01** and **sv02**) in an existing server pool (**sp01**), you could enter:

```
eqcli> srvpool sp01 si sv01,sv02
eqcli sp-sp01-si-sv01*>
```

When you enter multiple server instances as in the command above, eqcli enters a special combined context that applies commands to all of the specified objects. For example, after entering the example command above, eqcli enters the “`sv01,sv02`” context and the CLI prompt changes to include the first four letters of the combined context, “`sv01*`”. To display the full current context, use the following command:

```
eqcli sp-sp01-si-sv01*> context
The current context is: 'sv01,sv02'
eqcli sp-sp01-si-sv01*>
```

Using the no Form of a Command

Most commands that create objects and set parameters have a **no** form that you can use to delete an object or reset a parameter to its default value. The general format of the **no** command is:

```
no [keywords] {object|parameter}
```

The **no** keyword must be followed by a complete object context that specifies the object to delete or the parameter to reset:

- If the object or parameter is defined in the current context, then you do not need to specify any keywords.
- If the object or parameter is defined in a lower level context, then specify the appropriate contexts before the object or parameter name.

So, for example, type the following to delete cluster **cl00**:

```
eqcli > no cluster cl00
```

For objects and parameters that have lower object contexts (i.e., match rules, server instances, and subnets), you can use the no form at either the global context or in the lower object specific context:

```
eqcli > no cluster cl00 match ma00
eqcli > cluster cl00
eqcli cl-cl00> no match ma00
```

For parameters, the **no** form requires the complete command used to set the parameter, minus the argument setting the value. So, for example, to reset the value of the **resp** (responder) parameter on match rule **ma00** in cluster **cl00**, you can type any of the following:

```
eqcli > no cluster cl00 match ma00 resp
eqcli > cluster cl00
eqcli cl-cl00> no match ma00 resp
eqcli > cluster cl00 match ma00
eqcli cl-cl00-ma-ma00> no resp
```

The operation specified by the **no** form of a command takes effect immediately, even in explicit commit mode. In other words, a **no** command form never needs to be followed by a **commit**, **exit**, or **<ctrl-d>** command; it is committed to the configuration file immediately.

In all cases, the **no** form of a command always returns to the current context after completion.

Queued Commands

CLI commands that specify changes to the current configuration will either be committed to the configuration file as soon as they are entered, or queued to be committed using the **commit**, **exit**, or **<ctrl-d>** commands.

- If a *complete* command is executed for an object in a lower context, the command is committed to the configuration immediately. The current command context is not changed after the command is entered. A “complete” command is one that specifies all parameters required to add or modify the object.

For example, entering the following command to create a server creates the server immediately, and leaves eqcli in the global context:

```
eqcli > server sv01 proto tcp ip 192.168.0.210 port 80
eqcli: 12000287: Operation successful
eqcli >
```

- If an *incomplete* command is executed for an object in a lower context, the command is queued to be committed to the configuration until a **commit**, **exit**, or **<ctrl-d>** command is entered. The current command context changes to the context of the object argument of the incomplete command. An “incomplete" command is one does not include one or more parameters required to add or modify the object.

For example, if the server **sv01** does not exist, entering the following **server** command in the global context queues the command and leaves eqcli in the relevant context:

```
   eqcli > server sv01
   eqcli sv-sv01>
```

- If a command is entered that affects only the object associated with the current context, the command is queued to be committed to the configuration until a **commit**, **exit**, or **<ctrl-d>** command is entered. The current command context does not change.

For example, if **sv01** exists and the current context is “**sv-sv01**”, then the following commands are queued until a **commit**, **exit**, or **<ctrl-d>** command is entered:

```
   eqcli > server sv01
   eqcli sv-sv01> ip 192.168.0.211
   eqcli sv-sv01> port 8080
   eqcli sv-sv01> commit
```

Queued commands can be committed or discarded using the following commands:

- **commit -** Commits all queued commands; does not change the current context.
- **exit <ctrl-d> -** Commits all queued commands and changes to the next highest context in the hierarchy (if executed in the global context, either of these commands exits eqcli).
- **discard -**Discards all queued commands; does not change the current context.
- **quit -**Discards all queued commands and changes to the next highest context in the hierarchy (if executed in the global context, this command exits eqcli).

Note that the following commands always take effect immediately and do not change the current command context:

- A command that sets a global parameter (see Global Commands).
- The **no** form of a command (see Using the "no" Form of a Command.
- The **show** command in any context.

Context Help

You can type <?> in a number of situations to display context help:

- If you type **<?>** at the CLI prompt, a list of commands that are valid in the current context is displayed. For example, this command displays help for all global commands as shown in "Global Commands" on page 164:

```
eqcli >?
```

Entering the following two commands displays help for all the commands available in the **cluster cl01** specific context, as shown in "Cluster and Match Rule Commands" on page 171.

```
eqcli > cluster cl01
eqcli cl-cl01>?
```

- If you type the complete name of a command that is valid in the current context and type **<?>**, context help for that command is displayed. For example:

```
eqcli > cluster cl01
eqcli cl-cl01> clientto?
clientto: Set the client timeout for this cluster.

Syntax: cluster <cluster name> clientto <value>
Warning: Only valid for proto http or https.
```

- If you type a partial command name and type **<?>**:

If there is only one command that matches the string entered, context help for that command is displayed.

If there are multiple commands that begin with the string entered, the names of all the matching commands are displayed.

Global Parameters

Global or System Parameters include Probes and Networking. Most clusters will work with the default values on these tabs. To view or modify the default global parameter values:

1. Start the Equalizer CLI and log in.
2. Enter the following:

```
eqcli > show
```

The following will summary will be displayed that shows the global parameters that are configured on Equalizer. Refer to "Global Commands" on page 164 for descriptions of each parameter.

```
eqcli > show

Variable                Value

icmp_interval           15
icmp_maxtries           3
icmp_relax_probe        disabled
hostname                NAME
date                    Mon Jun 17 18:15:40 UTC 2013
timezone                UTC
locale                  en
global services         http, https, ssh, snmp, Envoy,
                        Envoy_agent
name-servers            None
ntp-server              pool.ntp.org - Unavailable: name-server undefined
syslog-server           None
alerts                  Enabled
extended audit          Enabled
GUI logo                Fortinet, Inc.
boot image              Equalizer Image A Version 10.2.1b (Build 23796)
boot image              Equalizer Image A SB/OS-10

eqcli >
```

Show Configuration Command

The show configuration command can be used to display all current configuration data from the CLI. Enter the following. The display shown is an abridged version of an actual output:

```
eqcli > show config
sequence = 60
locale = "en"
watchdog = 30
version = 3
extended_audit = true
customer {
  sequence = "0"
  # last_refresh_date = ""
  # support_email = ""
  # support_enddate = ""
  # hw_support_level = ""
  # fw_support_enddate = ""
  # fw_support_level = ""
  # en_support_enddate = ""
  # en_support_level = ""
}
```

```
}
ntp {
  sequence = "0"
  enable = true
  server = "pool.ntp.org"
}
syslog {
  sequence = "0"
  enable = false
  # server = ""
}
alerts {
  sequence = "0"
  enable = true
}
services {
  sequence = "0"
  http = true
  https = true
  ssh = true
  snmp = true
  envoy = true
  envoy_agent = true
  fo_http = true
fo_https = true
```

```
fo_ssh = true
fo_snmp = true
fo_envoy = true
fo_envoy_agent = true
}
```

Debug Commands

The debug mode can display hidden commands for the following functions and can be accessed using the CLI only. You can access it when booting your appliance and entering **CTRL+C** when prompted for a **username**. The following commands are available.

Debug Commands	
debug > **boot *image***	: *Set the EQ/OS image (A or B) to use at next boot.*
debug > **exit**	: *Halt the system.*
debug > **halt**	: *Halt the system.*
debug > **help**	: *Displays this text.*
debug > **login**	: *Log in as a system user using stored passwords.*
debug > **nodog**	: *Disable system watchdog timer for current boot.*
debug > **reboot**	: *Reboot the system.*
debug > **reset config**	: *Reset the configuration to factory defaults.*
debug > **reset for-support**	: *Reset the configuration to factory defaults. Keeps core files and files that are currently in the file store. A backup file is also generated and stored in the file store.*
debug > **reset keep-license**	: *Reset the configuration to factory defaults and retain the license data in the configuration.*
debug > **reset passwd**	: *Reset the 'touch' user password.*
debug > **shell**	: *Obtain non-privileged shell access. Requires one-time password from Support.*
debug > **shell admin**	: *Obtain privileged shell access.*
debug > **show boot**	: *Display the available EQ/OS boot images.*
debug > **version**	: *Displays the running system version information.*

Resetting Your Password

You can reset your password to the default by entering the following when your unit is rebooting:

1. Enter **CTRL+C** when prompted for a **username**. This will enter the debug mode.
2. Enter the following:

```
debug > reset passwd

Reset password successful.

debug >
```

3. Enter `exit` to return to eqcli login prompt.

Note - The "reset passwd" command may fail with the following error message:

```
cli_reset_passwd: 68400010: Could not complete 'get user' request.:
Connection refused
Unable to reset 'touch' password.
```

This indicates that the configuration management daemon is not available to process the command. If successive attempts to run this command fail with the above error message, contact technical support.

Resetting the Configuration

This command resets Equalizer configuration to a factory installed condition. All VLANs, subnets, clusters, servers, SSL certificates, and other user-supplied objects and settings will be removed. After the configuration has been reset, the system will be rebooted.

1. Enter **CTRL+C** when prompted for a **username**. This will enter the debug mode.
2. Enter the following:

```
debug > reset config

WARNING! This command resets the Equalizer configuration to a
factory installed condition. All VLANs, subnets, clusters, servers,
SSL certificates, and other user-supplied objects and settings will
be removed. After the configuration has been reset, the system will
be rebooted. Do you want to continue (Y/N)?
```

3. Select `Y` or `N`.
4. Enter `exit` to return to `eqcli` login prompt.

Using the reset keep-filestore Command

This command resets Equalizer configuration to a factory installed condition. All VLANs, subnets, clusters, servers, SSL certificates, and other user-supplied objects and settings will be removed. After the configuration has been reset, the system will be rebooted.

This command saves all of the files that are currently in the files store as well as the core files. A backup file will be generated and may be removed if necessary.

1. Enter **CTRL+C** when prompted for a **username.** This will enter the debug mode.
2. Enter the following:

```
eqcli hidden reset keep-filestore

WARNING! This command resets the Equalizer configuration to a
factory installed condition. All VLANs, subnets, clusters, servers,
SSL certificates, and other user-supplied objects and settings will
be removed. After the configuration has been reset, the system will
be rebooted. Do you want to continue (Y/N)?
```

3. Select `Y` or `N`.
4. Enter **`exit`** to return to `eqcli` login prompt.

Context Command Summaries

This section contains a table for each CLI context that summarizes all the commands that can be executed in each context. The following typographical conventions are used when describing command syntax and usage.

`eqcli>`	`Regular constant width type` is used for the eqcli command prompt and messages.
`eqcli>` **`vlan list`**	**`Bold constant width type`** is used for commands you type.
`eqcli> vlan` *`vlname`* **`show`**	***`Bold italic constant width type`*** is used for command elements that you must specify, such as an object name or a parameter value.
`{option | option...}`	A series of elements in braces ("{", "}"), separated by vertical bars ("|"), means you must choose one of the options between the braces. The braces are not typed on the command line.
`{option,option...}`	A series of elements in braces ("{", "}"), separated by commas (","), means you may chose more than one of the options between the braces. Separate multiple options on the command line using either commas or vertical bars. If you use white space in the string of options, the entire string must be surrounded by quotes. The braces are not typed on the command line.
`[option]`	Square brackets ("[", "]") indicate optional command elements. The brackets are not specified on the command line.
`eqcli vlan> *ip ip_addr`	An asterisk (*) before a parameter indicates that the parameter must be set before the associated object can be created.
`# Text in the right margin`	Text in italic font following the pound character (#) in the right margin is a comment indicating the purpose of the command and should not be typed onto the command line. Details appear in notes following each table.

Global Commands

The table below lists the global configuration commands that are available in the global context of the CLI. These commands allow you to:

- Configure, enable, and disable settings such as hostname, NTP, and DNS.
- Perform system operations, such as upgrading and rebooting.

Global Commands	
eqcli > **alerts**	: *Global Enable/Disable alerts.*
eqcli > **agr**	: *Add or modify an AGR or interface instance.*
eqcli > **backup**	: *Upload a system backup to remote FTP.*
eqcli > **boot**	: *Set the OS image (A or B) to use on at next boot.*
eqcli > **certificate**	: *Add or modify an SSL certificate.*
eqcli > **cluster**	: *Add or modify a cluster or a match rule.*
eqcli > **context**	: *Display the current command context.*
eqcli > **crl**	: *Add or modify a Certificate Revocation List (CRL).*
eqcli > **date**	: *Set the system time.* *Date [[[[[CC]yy]mm]dd]HH]MM[.SS], where cc = century; yy = last two digits of the current year (e.g., 11, 12, etc.); mm = the number of the month, specified as two digits; dd = the number of the day, specified as two digits; HH = the hour on which to filter, in two digits; MM = the minutes on which to filter, in two digits; required. ss = the number of seconds.*
eqcli > **diags**	: *Run the system utilities commands in the 'diags' context.*
eqcli > **edit**	: *Edit a file in the datastore.*
eqcli > **ext_services**	: *Add or modify a mail server in the 'ext_ services' context.*
eqcli > **exit**	: *Commit all pending configuration changes and exit eqcli.*
eqcli > **extended_audit**	: *Enable or disable extended audit logging.*
eqcli > **failover**	: *Enter 'failover' context.*
eqcli > **files download**	: *Download a file onto the Equalizer. <url> := url of file to download.*

Global Commands	
eqcli > **files edit**	: *Edit a datastore file.*
eqcli > **files ftp**	: ***file*** *is the file name.* ***server*** *- url of the FTP server onto which the file should be copied. ftp://[username:password@]hostname[/path]/*
eqcli > **firewall**	: *Enter firewall conext.*
eqcli > **forticare registration**	: *Load the Forticare registration information.*
eqcli > **geocluster *name parameter value***	: *Add or modify a GeoCluster or a GeoSite instance.*
eqcli > **geosite *name parameter value***	: *Add or modify a GeoSite.*
eqcli > **guilogo**	: *Change the GUI logo of the Equalizer.*
eqcli > **halt**	: *Shutdown Equalizer.*
eqcli > **health_check**	: *Change to the command context for the specified health check.*
eqcli > **hostname**	: *Set the system hostname.*
eqcli > **illb-grp**	: *Add or modify the illb group.*
eqcli > **llb-gw**	: *Add or modify the llb gateway.*
eqcli > **ollb-grp**	: *Add or modify the ollb group.*
eqcli > **interface**	: *Modify an interface.*
eqcli > **license upload**	: *Load an upgrade image.*
eqcli > **locale**	: *Set the locale of the system.* *<locale> := en | ja* *'en' - to set English locale* *'ja' - to set Japanese locale*
eqcli > **name-server**	: *Add a DNS name server entry. One IP address can be specified on the command line. A total of 3 IP addresses can be added. DNS is enabled as long as there is one entry in the list.* *primary : Add a primary name-server* *secondary: Add a secondary name-server* *tertiary : Add a tertiary name-server*
eqcli > **no**	: *Reset a parameter or delete an object.*
eqcli > **ntp**	: *Enable or disable NTP (without changing the NTP configuration).*
eqcli > **ntp-server**	: *Set the NTP server name.*
eqcli > **peer**	: *Add or modify a failover peer.*
eqcli > **ping**	: *Send ICMP or ICMPv6 ECHO_REQUEST packets to a host.*

Global Commands	
eqcli > **quit**	: *Discards the entered configuration changes and exits eqcli.*
eqcli > **rebalance**	: *Rebalance clusters among failover group members. Each cluster will be re-started on its 'preferred peer'.*
eqcli > **reboot**	: *Reboot Equalizer.*
eqcli > **remote-mgmt**	: *Set the remote management options.*
eqcli > **reputation**	: *Modify IP Reputation.*
eqcli > **resp**	: *Add or modify responders.*
eqcli > **restore**	: *Restore a system backup from remote FTP.*
eqcli > **run_script**	: *Run an eqcli command script.*
eqcli > **server**	: *Add or modify a server.*
eqcli > **services [!]http,[!]https, {!]ssh,[!]snmp**	: *Set the default GUI and SSH access. These settings apply if 'services' is not set in a VLAN configuration.*
eqcli > **show**	: *Display configuration information. With no arguments, displays global parameters. Otherwise, displays either a list of all objects of one type (for example, 'cluster', 'srvpool', 'vlan') or the configuration of a specific object.*
eqalic > **smart_control**	: *Add or modify a Smart Control.*
eqcli > **snmp**	: *Add SNMP parameters.*
eqcli > **srvpool**	: *Add or modify a server pool.*
eqcli > **sse**	: *Modify global server-side encryption parameters.*
eqcli > **stats**	: *Display global statistics.*
eqcli > **syslog**	: *Enable or disable remote logging.*
eqcli > **syslog-server**	: *Set the syslog server IP address*
eqcli > **timezone**	: *Set the system timezone.*
eqcli > **traceroute**	: *Trace the network path to a host using UDP packets.*
eqcli > **tunnel**	: *Set the tunnel.*
eqcli > **upgrade**	: *Load an upgrade image.*
eqcli > **user**	: *Create or modify a user object.*
eqcli > **version**	: *Show detailed system and version information.*
eqcli > **vlan**	: *Add or modify a VLAN or subnet.*

Certificate Commands

Each SSL certificate installed on Equalizer has a CLI context that provides commands for managing the certificate and its associated private key. Certificates, private keys, and CRLs (see the following section) are used by Equalizer to provide SSL offloading for HTTPS clusters.

In SSL offloading, Equalizer terminates the SSL connection with the client, decrypts the client request using a certificate and key, sends the request on to the appropriate server, and encrypts the server response before forwarding it on to the client.

Certificates are uploaded to Equalizer and then associated with one or more clusters. Two types of certificates may be used to authenticate HTTPS cluster connections:

- A cluster certificate is required to authenticate the cluster to the client and to decrypt the client request (these are also called server certificates). For cluster certificates, both a certificate file and a private key file must be uploaded to Equalizer.
- A cluster may also be configured to ask for, or require, a *client certificate* -- a certificate used to authenticate the client to Equalizer. For client certificates, only a certificate file is uploaded to Equalizer(no keyfile is used).

Supported certificate commands are shown in the following tables.

Using Certificate Commands in Global Context	
eqcli > **certificate *certname* [*cmd* ...]**	: *Create certname (**req_cmds** = * commands below)*
eqcli > **certificate *certname cmd* ...**	: *Modify certname (**cmd** = any commands below)*
eqcli > **no certificate certname**	: *Delete **certname***
eqcli > **show certificate [*certname*]**	: *Display all certificates or **certname***
eqcli > **certificate *certname***	: *Change to "cert-certname" context (see below)*

Using Certificate Commands in Certificate Context		
eqcli cert-*certname*> **certfile {edit|*url*}**	:	*Upload SSL certificate* *edit ::= invokes an editor to enter the certificate content.* *url ::= fetches the certificate from the specified URL, which must point to a file on a FTP or HTTP server.* *Example:* *certfile edit* *certfile* *ftp://[<username>[:<password>]@] www.example.com/certfile.pem* *certfile http://www.example.com/certfile.pem*
eqcli cert-*certname*> **genscr**	:	*Create a CSR.*
eqcli cert-certname> **keyfile (edit|url)**	:	*Upload private key* *edit ::= invokes an editor to enter the private key content.* *url ::= fetches the private key from the specified URL, which must point to a file on a FTP or HTTP server.* *Example:* *keyfile edit* *keyfile* *ftp://[<username>[:<password>]@] www.example.com/keyfile.pem* *keyfile http://www.example.com/keyfile.pem*
eqcli cert-*certname*> **show**	:	*Display the certificate configuration.*
eqcli cert-*certname*> **signcsr** {**days|force**}	:	*Create a self-signed certificate.* *days ::= State name (SN) for the CSR.* *force ::= Force overwrite of existing self-signed certificate.*

The arguments to the **certfile** and **keyfile** commands are:

edit - Launch an editor to supply the content of the certificate or key file.

url - Download the certificate or key file using **ftp://** or **http://** protocol.

Using Certificate Commands in CSR Context

Using Certificate Commands in CSR Context		
eqcli cert-*certname*-csr > **common**	:	*Common Name (CN) for the CSR (Required).*
eqcli cert-*certname*-csr > **country**	:	*Country name (C) for the CSR (Default is 'US').*
eqcli cert-certname-csr > **force**	:	*Force overwrite of existing CSR.*
eqcli cert-*certname*-csr > **keylen**	:	*The length for the generated private key (Default is 2048).*
eqcli cert-*certname*-csr > **locality**	:	*Locality (L) for the CSR.*
eqcli cert-*certname*-csr > **method**		*Encryption method to be used for the private key (Default is 'rsa').*
eqcli cert-*certname*-csr > **org**		*Organization (O) for the CSR.*
eqcli cert-*certname*-csr > **orgunit**		*Organizational Unit (OU) for the CSR.*
eqcli cert-*certname*-csr > **state**		*State name (SN) for the CSR.*

Certificate Revocation List Commands

The **crl** context provides commands for managing Certificate Revocation Lists (or CRLs). CRLs can be used to verify that the certificates used by Equalizer are valid and have not been compromised. A CRL is uploaded to Equalizer using commands in the **crl** context, and then associated with one or more clusters in the cluster specific context. Whenever a certificate is used to authenticate a connection to the cluster, the CRL is checked to make sure the certificate being used has not been revoked. The supported commands in the **crl** context are shown in the following tables.

Note - If a CRL attached to a cluster was generated by a Certificate Authority (CA) different from the CA used to generate a client certificate presented when connecting to the cluster, an error occurs. The CRL and client certificate must be signed by the same CA.

Using CRL Commands in the Global Context	
`eqcli > certificate certname [cmd ...]`	: *Create certname (**req_cmds** = * commands below)*
`eqcli > certificate certname cmd ...`	: *Modify **certname (cmd** = any commands below)*
`eqcli > no certificate certname`	: *Delete **certname***
`eqcli > show certificate [certname]`	: *Display all certificates or **certname***
`eqcli > certificate certname`	: *Change to "cert-certname" context (see below)*

Using CRL Commands in a CRL specific Context	
`eqcli crl-crlname> crlfile {edit|url}`	: *Upload the CRL*
`eqcli crl-crlname> show`	: *Display CRL crlname*

The arguments to the **crlfile** command are:

- **edit -** Launch an editor to supply the content of the CRL file.
- **url -** Download the CRL file from the **ftp://** or **http://** protocol URL supplied on the command line.

Cluster and Match Rule Commands

Each cluster has its own context and the settings available in the cluster's context depends on the cluster's **proto** parameter -- this parameter must be specified first on the command line when creating a cluster. A Layer 7 cluster may have one or more match rules associated with it, each with its own context. Cluster and match rule commands are summarized in the tables below.

Using Cluster Commands in the Global Context

eqcli > **cluster *clname* *req_cmds***	: *Create **clname** (**req_cmds** = * commands below)*
eqcli > **cluster *clname* cmds ...**	: *Modify **clname** (**cmds** = any commands below)*
eqcli > **no cluster *clname***	: *Delete **clname***
eqcli > **show cluster [*clname*]**	: *Display all clusters or **clname***
eqcli > **cluster *clname***	: *Change to the "cl-clname" context(see below)*

Using Cluster Commands in a Cluster Specific Context

For all Clusters:	
eqcli cl-*clname*> ***ip *ip_addr***	: *Cluster IP address*
eqcli cl-*clname*> ***proto *{http|https|tcp|udp}***	: *Protocol --* **MUST SET proto FIRST**
eqcli cl-*clname*> ***port *integer***	: *Cluster port*
eqcli cl-*clname*> **show**	: *Show the cluster configuration*
eqcli cl-*clname*> **stats**	: *Display cluster statistics*
For Layer 7Clusters:	
eqcli cl-*clname*> **age *integer***	: *Cookie age in seconds (0 [default] to 31536000 -- one year)*
eqcli cl-*clname*> **clientto *integer***	: *Client connection timeout*
eqcli cl-*clname*> **compress_min *integer***	: *Set the minimum file size for compression in bytes when compression is enabled for Layer HTTP and HTTPS clusters. Valid values range from 0 to 1073741824 bytes. The default is 1024 bytes.*
eqcli cl-*clname*> **compress_types *string***	: *Mime types to compress*
eqcli cl-*clname*> **connto *integer***	: *Server connection timeout*

Using Cluster Commands in a Cluster Specific Context	
`eqcli cl-`*`clname`*`>` **`custhdr`** ***`string`***	*: Custom request header*
`eqcli cl-`*`clname`*`>` **`domain`** ***`string`***	*: Cookie domain*
`eqcli cl-`*`clname`*`>` **`flags`**	*: Disable and enable flags*
For Layer 7 Http Clusters: **`{[!]always,[!]compress, [!]disable,[!}allow_utf8 [!]ignore_case,[!]insert_client_ip, [!]no_header_rewrite, [!]once_only, [!]spoof,[!]tcp_mux}`**	
For Layer 7 https clusters: **`{[!]ics [!]allow_sslv2, [!]allow_sslv3,[!]push_client_cert, [!]require_client_cert,[!]rewrite_redirects, [!]strict_crl_chain,[!]ignore_critical_extns, [!]software_ssl_only,[!]allow_tls10, [!]allow_tls11 [!]allow tls12}`**	
`eqcli cl-`*`clname`*`>` **`gen`** ***`integer`***	*: Cookie generation (0 to 65535).*
`eqcli cl-clname >` **`hdredit`** ***`edit|url`***	*: Set the header edit expression for a cluster.* *<mode> ::= edit | <url>* *edit ::= invokes an editor to enter the desired header edit script.* *url ::= fetches the header edit script from the entered fully qualified ftp/http site or from the file store with the path to the header edit script file starting with 'file://'.* *Note: Only valid when cluster 'proto' = 'http' or 'https'.*
`eqcli cl-`*`clname`*`>` **`match maname`**	*: Change to the* ***maname*** *match context*
`eqcli cl-`*`clname`*`>` **`match`** ***`cmds`***	*: Execute match commands*
`eqcli cl-`*`clname`*`>` **`no match`** ***`maname`***	*: Delete match* ***maname***
`eqcli cl-`*`clname`*`>` **`no {age|clientto|connto |custhdr|domain |gen|path|resp|scheme |serverto|srvpool}`**	*: Reset the specified parameter*
`eqcli cl-`*`clname`*`>` **`path`** ***`string`***	*: Cookie path*
`eqcli cl-`*`clname`*`>` **`range`**	*: Set the cluster port range.*

Using Cluster Commands in a Cluster Specific Context

`eqcli cl-clname> resp rname`	`: Responder name`
`eqcli cl-clname> scheme integer`	`: Cookie scheme (0,1,2)`
`eqcli cl-clname> serverto integer`	`: Server response timeout`
`eqcli cl-clname> sni sni-name`	`: Server Name Indication name`
`eqcli cl-clname> sni-name sni_ svname servername`	`: Add the server name or list of server names in sni.`
`eqcli cl-clname> srvpool spname`	`: Server pool name`
`eqcli cl-clname> stats`	`: Display the statistics for cluster`
`eqcli cl-clname> staleto`	`: Set the stale timeout for a cluster.`
`eqcli cl-clname> stickyto`	`: Set the sticky timeout for a cluster.`
`eqcli cl-clname> stickynetmask`	`: Set the sticky netmask for a cluster.`
`eqcli cl-clname> cipherspec {url|edit|enter}`	`: Set the cipherspec for an HTTPS cluster.`
`eqcli cl-clname> clientca certname`	`: Attach a client certificate to an HTTPS cluster.`
`eqcli cl-clname> clflags {[!]allow_sslv2,[!]allow_sslv3, [!]push_client_cert,[!]require_client_cert, [!]strict_crl_chain}`	
`eqcli cl-clname> crl crlname`	
`eqcli cl-clname> no {cert|cipherspec |clientca|crl|valdepth}`	`: Reset the parameter to its default value`
`eqcli cl-clname> valdepth}`	`: Set validation depth for cluster.`
`eqcli cl-clname> preferred_peer`	`: Set the preferred peer`
`eqcli cl-clname> persist type {[!]none,[!]source_ip, [!]Coyote_cookie_0, [!]Coyote_cookie_1, !]Coyote_cookie_2`	`: Set the persist type`

For Layer 7 TCP Clusters (proto = tcp):

```
eqcli cl-clname> flags
     {[!]disable,[!]spoof,
      [!]delayed_binding,[!]abort_server,
      [!]ics
```

Using Cluster Commands in a Cluster Specific Context

eqcli cl-*clname*> **stickyto**	: *Set the sticky timeout for a cluster.*
eqcli cl-*clname*> **stickynetmask**	: *Set the sticky netmask for a cluster.*
eqcli cl-*clname*> **srvpool** ***spname***	: *Server pool name*
eqcli cl-*clname*> **preferred_peer**	: *Set the preferred peer*
eqcli cl-*clname*> **clientto** ***integer***	: *Client connection timeout*
eqcli cl-*clname*> **serverto** ***integer***	: *Server response timeout*
eqcli cl-*clname*> **connto** ***integer***	: *Server connection timeout*
For Layer 4 Clusters (proto = tcp or udp):	
eqcli cl-*clname*> eqcli cl-clname> **flags** **{[!]dsr,[!]ics!]spoof, [!]disable}**	
eqcli cl-*clname*> **idleto** ***integer***	: *Set the connection idle timeout*
eqcli cl-*clname*> **no** **{idleto|stickyto**	: *Reset specified parameter to default value*
eqcli cl-*clname*> **range** ***integer***	: *Upper limit of port range*
eqcli cl-*clname*> **stickyto** ***integer***	: *Set the connection sticky timeout*
eqcli cl-*clname*> **stickynetmask**	: *Set the sticky netmask for a cluster.*

Note - Match Rules are not supported on E250GX model Equalizers.

Using Match Rule Commands in the Global Context

eqcli > **cluster** ***clname*** **match** ***maname*** ***req_cmds***	: *Create* **maname** ***(req_cmds*** *= * commands below)*
eqcli > **cluster** ***clname*** **match** ***maname*** ***cmd*** ...	: *Modify* **maname** ***(cmds*** *= any commands below)*
eqcli > **no cluster** ***clname*** **match** ***maname***	: *Delete match rule* **maname**
eqcli > **show cluster** [***clname***]	: *isplay all match rules or* **maname**
eqcli > **cluster** ***clname*** **match** ***maname***	: *Change context to a match rule context*

Using Match Rule Commands in a Match Rule Specific Context	
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`{disable|enable}`**	: *Disable match rule*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`age integer`**	: *Cookie age in seconds (0 to 31536000 -- one year)*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`compress_min`** ***`integer`***	: *Minimum bytes to compress (0 to 1073741824 -- default 1024)*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`compress_types`** ***`string`***	: *Mime types to compress*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`domain`** ***`string`***	: *Cookie domain*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`expression`** ***`string`***	: *Match expression*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`flags`**	: *Enable/disable Flags*
`[!]abort_server,[!]always, [!]client_ip,[!]compress, [!]ignore_case,[!]no_header_rewrite, [!]once_only,[!]persist, [!]spoof,[!]tcp_mux}`	:
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`gen integer`**	: *Cookie generation (0 to 65535)*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`hredit`** ***`edit|url`***	: *Set the match rule header edit expression.* *edit ::= invokes an editor to enter the desired header edit script.* *url ::= fetches the header edit script from the entered fully qualified ftp/http site or from the file store with the path to the header edit script file starting with 'file://'.*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`*nextmatch`** ***`maname`***	: *Next match in list*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`no {age|domain|expression|gen |path|resp|scheme|srvpool}`**	: *Reset parameter*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`path`** ***`string`***	: *Cookie path*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`resp`** ***`rname`***	: *Set Responder name*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`scheme integer`**	: *Cookie scheme (0, 1, 2)*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`show`**	: *Show configuration*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`srvpool`** ***`spname`***	: *Server Pool name*
`eqcli cl-`*`clname`*`-ma-`*`maname`*`>` **`stats`**	: *Display statistics*

Cluster and Match Rule Command Notes

- When creating a cluster, the list of available parameters depends on the protocol selected for the cluster. As a result, the **proto** parameter must be specified before any other cluster parameters on the command line.
- Layer 7 clusters can have one or more match rules that override the options set on the cluster when the expression specified in the match rule matches an incoming client request. (Layer 4 clusters do *not* support match rules.)
- The cluster flags supported for a particular cluster depend on the setting of the cluster **proto** parameter, as shown in the table below.

Cluster Flags

A flag may be turned off by prefixing with "!".

Cluster 'proto'	Flag	Description
tcp and udp	dsr	Enables “direct server return" -- servers respond directly to clients rather than through Equalizer.
	ics	Enables “inter-cluster sticky” -- Layer 4 persistence is preserved across clusters and server ports.
	spoof	Disables Source NAT (SNAT) -- the client IP address is used as the source IP in packets sent to servers.
http and https	abort_server	Close server connections without waiting.
	always	Always insert a cookie into server responses.
	client_ip	Include the client IP address in headers.
	compress	Compress server responses.(Not valid for E250GX)
	ignore_case	Do not consider case when evaluating a match rule.
	no_header_rewrite	Do not rewrite Location headers in server responses.
	once_only	Evaluate the first set of headers in a client connection only.
	persist	Insert a cookie in server responses if the server did not.
	spoof	Use the client IP as source IP in packets sent to servers.
	tcp_mux	Enables TCP multiplexing for a cluster. TCP multiplexing must also be enabled on at least one server instance in the server pool assigned to the cluster (or one of its match rules). See the section.
https only	allow_sslv2	Enable SSLv2 for client connections.
	allow_sslv3	Enable SSLv3 for client connections. This option is enabled by default.

Cluster 'proto'	Flag	Description
	push_client_cert	Send the entire client certificate to the back-end server. This allows the server to confirm that the client connection is authenticated without having to do a complete SSL renegotiation.
	require_client_cert	Require that clients present certificates.
	software_ssl_only	This flag appears only on systems that are equipped with Hardware SSL Acceleration. When enabled, it specifies that all SSL operations will be performed in software, instead of being performed using the SSL accelerator hardware. This flag does not appear on systems that are not equipped with Hardware SSL Acceleration, since on these units SSL operations are always performed in software. This flag is disabled by default. All units with Hardware SSL Acceleration can process the TLSv1.0, TLSv1.1, and TLSv1.2 protocols in both hardware and software, except for legacy GX hardware. On legacy GX hardware, only TLSv1.0 is supported by Hardware SSL Acceleration; if you want to enable TLSv1.1 or TLSv1.2 on GX hardware, you must first enable this flag. Please note that enabling this option will reduce the processor and memory resources generally available for processing cluster traffic, since performing SSL operations in software requires use of the system CPU and system memory (instead of the dedicated SSL acceleration hardware CPU and memory).
	allow_tls10	This option enables and disables support for the TLSv1.0 protocol. Enabled by default. If multiple TLS versions are enabled, the first supported TLS version negotiated by a client will be used.
	allow_tls11	This option enables and disables support for the TLSv1.1 protocol. Disabled by default. If multiple TLS versions are enabled, the first supported TLS version negotiated by a client will be used.
	allow_tls12	This option enables and disables support for the TLSv1.1 protocol. Disabled by default. If multiple TLS versions are enabled, the first supported TLS version negotiated by a client will be used.

Cluster 'proto'	Flag	Description
	rewrite_redirects	When enabled, forces Equalizer to pass responses from an HTTPS cluster’s servers without rewriting them. In the typical Equalizer setup, you configure servers in an HTTPS cluster to listen and respond using HTTP; Equalizer communicates with the clients using SSL. If a server sends an HTTP redirect using the Location: header, this URL most likely will not include the https: protocol. Equalizer rewrites responses from the server so that they are HTTPS. You can direct Equalizer to pass responses from the server without rewriting them by enabling this option.
	ignore_critical_extns	Control whether Equalizer will process "CRL Distribution Point" extensions in client certificates. This option only affects the processing of the "CRL Distribution Point" extension in client certificates: When **Ignore Critical Extensions** is disabled, a client certificate presented to Equalizer that includes any extension will be rejected by Equalizer. This is the behavior in previous releases. When **Ignore Critical Extensions** is enabled (the default), a client certificate presented to Equalizer that has a CRL Distribution Point extension will be processed and the CRL critical extension will be ignored. Note, however, that if other extensions are present in a client certificate they are not ignored and will cause the client certificate to be rejected by Equalizer.
	strict_crl_chain	Check the validity of all certificates in a certificate chain against the CRL associated with the cluster. If any of the certificates in the chain cannot be validated, return an error. If this option is *disabled* (the default), only the last certificate in the chain is checked for validity.

Diagnostic Commands

Using Diagnostic Commands in a Global Context	
eqcli > **diags arp**	: *Display the ARP entries.*
eqcli diags > **cache srvpool *sp_name* id *id* flush**	: *Flush a single entry from the server pool cache using an ID (obtained by displaying all cache entries).*
eqcli diags > **cache srvpool *sp_spname* flush**	: *Flush all entries from a server pool cache.*
eqcli diags > **cache flush**	: *Flush all entries from all server pool caches.*
eqcli diags > **cache srvpool *srvplname* flush**	: *Deletes all the cache entries of a particular server pool*
eqcli diags > **show cache srvpool *srvplname***	: *Display all the cache entries of a particular server pool*
eqcli diags > **context**	: *Displays the current command context.*
eqcli > **diags df**	: *Display the disk space on the file system.*
eqcli > **diags dig server *ip_address* hostname *hostname***	: *Display the DNS look up information.*
eqcli > **diags eqcollect**	: *Download eqcollect on the local filesystem or upload it to a FTP server.* *local: Perform the eqcollect on the local system.* *name : Specify the file name to be embedded in eqcollect file name as an identifier.* *url : Specify the URL for the eqcollect.*
eqcli > **diags exit**	: *Commit all pending configuration changes and change to the global context.*
eqcli > **diags ifconfig**	: *Display the state of all interfaces.*
eqcli > **diags ipmi**	: *Issue a power on, power off, shutdown or power reset command to a server.*
eqcli > **diags netstat**	: *Display the network status information.*
eqcli > **diags ps**	: *Display the information about all the processes.*
eqcli > **diags quit**	: *Discard all pending configuration changes and change to the global context.*
eqcli > **diags top**	: *Display the top processes on the system.*
eqcli > **diags tcpdump fg count *pkt_count* capture |server *server_name* |cluster *cluster_name* | iface *iface_name* | vlan *vlan_name* | agr *agr_name* expr *expression***	: *Save the description of the content of packets on the network.* *(i.e. tcpdump fg count 100 capture vlan vl01 expr "tcp[13] = 2")*

External Services Commands

Using External Services Commands in the Global Context	
eqcli > **ext_services**	*: Add or modify a mail server in the'ext_services' context.*
eqcli > **show ext_services**	*: Display the configured external services.*

External Services Context Commands	
eqcli *xs*> **no** ***smtp_relay name***	*: Delete the specified SMTP Relay mail server.*
eqcli *xs*> **show** ***smtp_relay name***	*: Display a list of SMTP Relay mail servers, or detail for the specified SMTP Relay mail server.*
eqcli *xs*> **smtp_relay** ***name***	*: Add or modify a SMTP Relay (mail server).*
eqcli *xs*> **no** ***vlb_manager name***	*: Delete the specified VLB Manager.*
eqcli *xs*> **show** ***vlb_manager name***	*: Display a list of VLB Managers*
eqcli *xs*> **vlb_manager** ***name***	*: Add or modify a VLB Manager.*

Using SMTP Relay Commands in SMTP Relay Context	
eqcli xs-smtp-*smtpname* > ***Port***	*: Set the SMTP mail server port*
eqcli xs-smtp-*smtpname* > ***Server***	*: Set the SMTP mail server IP address. Required.*

Using VLB Manager Commands in VLB Manager Context

```
eqcli xs-vlb-vlbmgrname > flags
    {[!]disable}

eqcli xs-vlb-vlbmgrname > password       : Set the password for
                                           authenticating a user.

eqcli xs-vlb-vlbmgrname > timeout        : Set number of elapsed seconds
                                           for connection timeout.

eqcli xs-vlb-vlbmgrname > url            : Set the URL used to connect to
                                           the VLB Manager.

eqcli xs-vlb-vlbmgrname > username       : Set the user name for
                                           authenticating a user.
```

Failover Commands

The table below lists the failover global configuration commands that are available in the global context of the CLI.

Global Commands	
eqcli > **commit**	: *Commit all pending alert configuration changes.*
eqcli > **context**	*Display the current command context.*
eqcli > **cmd_subnet** ***vlanname:subnetname***	*Set the designated vlan:subnet as the command subnet.*
eqcli > **exit**	*Commit all pending alert configuration changes and exit to the user context.*
eqcli > **rebalance**	*For A/A Failover configurations = Rebalance clusters among failover group members. Each cluster will be re-started on its 'preferred peer'.* *For A/P Failover configurations = if the preferred_primary peer is in Backup mode and the rebalance command is used, the preferred_ primary will become Primary.*
eqcli > **quit**	*Discard all pending alert configuration changes and exit to the user context.*
eqcli > **show failover**	*Display the failover details.*

Firewall Commands

When you create a subnet, IP Filter (firewall) rules are automatically generated.The Firewall commands can disable these rules that may be used for troubleshooting or diagnostic purposes.

Disabling the firewall turns off all system packet filtering . Any subnet permit/deny rules are ignored and all traffic will be routed between subnets.

Firewall Commands in Global Context	
eqcli > **firewall context**	: *Display the current command context.*
eqcli > **firewall disable**	: *Disable system firewall.*
eqcli > **firewall enable**	: *Enable system firewall.*
eqcli > **firewall exit**	: *Commit all pending configuration changes and change to the global context.*
eqcli > **firewall quit**	: *Discard all pending configuration changes and change to the global context.*
eqcli > **firewall show**	: *Show firewall configuration*

Firewall Commands in Firewall Context	
eqcli fw> **commit**	: *Commit all pending configuration changes and do not change context.*

Firewall Commands in Firewall Context	
eqcli fw> **context**	: *Display the current command context.*
eqcli fw> **disable**	: *Disable system firewall.*
eqcli fw> **enable**	: *Enable system firewall.*
eqcli fw> **exit**	: *Enable system firewall. Commit all pending configuration changes and change to the global context.*
eqcli fw> **quit**	*Discard all pending configuration changes and change to the global context.*
eqcli fw> **show**	*Show firewall configuration.*

GeoCluster and GeoSite Instance Commands

Envoy provides cluster load balancing between Equalizers running at two or more geographically distributed locations -- called GeoSites. Each GeoSite is configured with a cluster that is capable of responding to requests for the same content. A GeoCluster is a collection of GeoSites that act together to determine the "best" GeoSite to respond to a particular request.

Envoy works together with special entries in the Domain Name System (DNS) configuration of the authoritative name server for a website.

Both GeoClusters and GeoSites are top-level objects in the CLI. In general:

1. Create a GeoCluster for your website (see below).
2. Create GeoSites
3. Add GeoSite Instances to GeoClusters (see below).

Using GeoCluster Commands in the Global Context	
eqcli > **geocluster *gcname req_cmds***	: *Create geocluster (see below for **cmds**)*
eqcli > **geocluster *gcname cmds***	: *Modify geocluster (see below for **cmds**)*
eqcli > **no geocluster *gcname***	: *Delete geocluster*
eqcli > **show geocluster**	: *Display geocluster summary*
eqcli > **show geocluster *gcname***	: *Display geocluster details*
eqcli > **geocluster *gcname***	: *Change context to "gcl-gcname"*

GeoCluster Context Commands	
eqcli *gcl-gclname*> **flags {[!]icmp}**	: *geocluster flags*
eqcli *gcl-gclname*> **fqdn*fqdn* *name***	: *FQDN for the geocluster website*
eqcli *gcl-gclname*> **gsi *gsiname***	: *Change to the geosite instance context*
eqcli *gcl-gclname*> **gsi *gsiname cmds***	: *Execute geosite instance commands*
eqcli *gcl-gclname*> **mx**	: *Set the GeoCluster Mail Exchange FQDN.*
eqcli *gcl-gclname*> **mrmax**	: *Maximum number of allowable resource records that will be returned in a DNS response*
eqcli *gcl-gclname*> **policy *policy***	: *GeoCluster Load Balancing policy*
eqcli *gcl-gclname*> **stats**	: *Display the statistics for the GeoCluster.*
eqcli *gcl-gclname*> **respv *integer***	: *Load balancing policy responsiveness*

GeoCluster Context Commands	
eqcli *gcl-gclname>* **ttl *integer***	: *DNS cache lifetime for Envoy responses*

Using Geosite Instance Commands in the Global Context	
eqcli > **geocluster gclname *gsi gsiname req_cmds***	: *Create a geosite instance*
eqcli > **geocluster gclname gsi *gsiname cmds***	: *Modify a geosite instance*
eqcli > **no geocluster gclname *gsi gsimaname***	: *Delete a geosite instance*
eqcli > **show geocluster gsi**	: *Display geosite instance summary*
eqcli > **show geocluster *gclname* gsi**	: *Display geosite instance details*
eqcli > **cluster *clname* match *maname***	: *Change to geosite instance context*

Geosite Instance Context Commands	
eqcli gcl-*gclname*-gsi-*gsiname*> ***load_weight***	: *GeoSite Instance weight (0-200)*
eqcli gcl-*gclname*-gsi-*gsiname*> **flags [!]default,[!]disable, [!]hot_spare,[!]preferred}**	

GeoCluster **flags** can be either **icmp** (enable ICMP triangulation) or **autof** (automatic fallback). [The **autof** option is not yet implemented.]

Geosite Instance Flags

A flag may be turned off by prefixing with "!".

default	When enabled, designates this GeoSite instance as the default GeoSite instance for the GeoCluster. Envoy load balances to the default GeoSite instance whenever it cannot choose a GeoSite instance based on probe responses. [This can happen, for example, when probe responses are not received from any site, when the resource (cluster) is down at all available sites, etc.] If no default GeoSite instance is selected for a GeoCluster and all GeoSites are down, then Envoy sends a null response to the client DNS.
disabled	When enabled, this GeoSite instance will not be selected as a response to a DNS query.
hot_spare -	When enabled, indicates that this GeoSite instance will be selected only when no other sites are available.
preferred	When enabled, indicates that this GeoSite instance will always be selected if it is available.

GeoSite and GeoSite Resource Commands

A GeoSite definition points to running Envoy and a cluster defined on that a GeoCluster defined on the Equalizer. GeoSites are associated with GeoClusters by using the GeoSite name when creating a GeoSite Instance. See "GeoCluster and GeoSite Instance Commands" on page 184.

Using GeoSite Commands in the Global Context	
eqcli > **geosite** ***gsname req_cmds***	: *Create geosite (see below for* ***req_ cmds)***
eqcli > **geosite** ***gsname cmds***	: *Modify geosite (see below for* ***cmds****)*
eqcli > **no geosite** ***gsname***	: *Delete geosite*
eqcli > **show geosite** ***gsname***	: *Display one or all geosites*
eqcli > **geosite** ***gsname***	: *Change to geosite instance context*

GeoSite Commands in GeoSite Context	
eqcli gs-*gsname*> **address** ***addr[,addr]***	: *GeoSite address (max: 1 IPv4 and 1 IPv6)*
eqcli gs-*gsname*> **agent** ***addr***	: *IP address of Envoy site*
eqcli gs-*gsname*> **resource** ***clname***	: *Cluster name at GeoSite*

GeoSite Commands in the GeoSite Context	
eqcli gs-*gsname*> **agent *addr***	: *Set the agent IP address for a GeoSite.*
eqcli gs-*gsname*> **commit**	: *Commit all pending GeoSite configuration changes and do not change context.*
eqcli gs-*gsname*> **context**	: *Display the current command context.*
eqcli gs-*gsname*> **exit**	: *Commit all pending GeoSite configuration changes and change to the global context.*
eqcli gs-*gsname*> **no*resource name***	: *Delete a resource.*
eqcli gs-*gsname*> **quit**	: *Discard all pending GeoSite configuration changes and change to the global context.*
eqcli gs-*gsname*> **resource *resource name***	: *Create a resource or change to the command context for the specified resource.*
eqcli gs-*gsname*> **show**	: *Display the GeoSite configuration, list all resources, or display details for the specified resource.*
eqcli gs-*gsname*> **type** ***[remote]*** ***[local]***	: *Set the type for this geosite. GeoSite agent is located on a remote machine. GeoSite agent is located on this local machine.*

GeoSite Resource Commands in the GeoSite Resource Context	
eqcli gs-*gsname*-rsrc-***resource name***> ***healthchk***	: *Attach one or more health checks to this resource.*

Health Check Commands

Health checks to determine the health of your servers and applications is an important benefit provided by Equalizer. Without health checking, a client may send a request to a "dead" server without knowing it. An administrator would need to manually intervene to replace the server.

With Equalizer, health checks can be attached to servers, server instances, server pools, geosite resources, and LLB gateways. They assist Equalizer in making server load balancing decisions.

Supported health check commands are shown in the following tables.

Refer to "Health Checks" on page 549 for detailed descriptions of the health check configuration.

Using Health Check Commands in a Global Context	
eqcli > **health_check *health_check_name* type *hctype*** **{acv|l3|tcp|udp|srvag|vlb}**	: *Create a health_check(commands below)*
eqcli > **health_check *health_check_name* *cmds***	: *Modify health_check(cmds = any commands below)*
eqcli > **no health_check *health_check_ name***	: *Delete health_check*
eqcli > **show health_check *health_check_ name***	: *Display all health check for health_check_name.*
eqcli > **health_check *health_check_name***	: *Change to the "hc-hcname" context (see below)*

Using Health Check Commands in Health Check Context	
eqcli-hc-*hcname* > **commit**	: *Commit all pending health check configuration changes.*
eqcli-hc-*hcname* > **context**	: *Display the current command context.*
eqcli-hc-*hcname* > **exit**	: *Commit all pending health check configuration changes and exit to the top level context.*

Using Health Check Commands in Health Check Context		
eqcli-hc-*hcname* > **flags**	:	*Set the health check flags.*
For All health checks **{disable}**		*disable = disables the health check on a global level.*
For ICMP and UDP health checks **{[!]relaxed_probe}**		*relaxed_probe (ICMP) = When enabled, if a server is DOWN, but has not previously been UP, it will be marked UP. Setting of this flag prevents the sudden reporting of servers as being DOWN following an upgrade.*
		relaxed_probe (UDP) = Determines what happens when no response is received to a UDP health check probe. If this flag is enabled, the object is marked UP. If this flag is disabled, the object is marked DOWN.
For ACV , L4 TCP, and Server Agent health checks **{[!]probe_ssl,[!]highest_tls_ version}**		*probe_ssl= If enabled, the probe exchange between Equalizer and a server instance will be performed over an encrypted SSL connection.*
		*highest_tls_version = Set the health check highest TLS version. Used when the probe_ssl option is enabled. Sets the health check highest TLS version. It specifies the highest TLS version that will be offered in the SSL probe sent to servers The probe can use levels from **SSLv3** and the highest probe levels of **TLS 1.0**, **1.1**, or **1.2***
		Valid values are:
		1 = SSL v3
		2 = TLS v1
		3 = TLS v1.1
		4 = TLS v1.2
		(Default: 4)
*eqcli-hc-hcname > **heavy_load value***	:	*Set the heavy_load value for the health check. 'heavy_load' is a floating-point value.*
eqcli-hc-*hcname* > **ip *IP address***	:	*Set the server IP address*
eqcli-hc-*hcname* > **light_load *value***	:	*Set the light_load value for the health check. 'light_load' is a floating-point value.*
eqcli-hc-*hcname* > **no**	:	*Delete a health check or reset a health check parameter to its default value.*

Using Health Check Commands in Health Check Context	
eqcli-hc-*hcname* > **probe_cto** ***Connection timeout***	: *Set the health check probe connect timeout (in seconds).*
eqcli-hc-*hcname* > **probe_dto** ***data timeout***	: *Set the health check probe data timeout (in seconds).*
eqcli-hc-*hcname* > **probe_gto** ***global timeout***	: *Set the health check probe global timeout (in seconds).*
eqcli-hc-*hcname* > **probe_interval** ***probe interval***	: *Set the interval between health check probes (in seconds).*
eqcli-hc-*hcname* > **probe_maxtries** ***max tries per interval***	: *Set the maximum number of attempts per interval before marking a server 'down'.*
eqcli-hc-*hcname* > **port** ***port number***	: *Set the port number for probing the server.*
eqcli-hc-*hcname* > **query** ***query***	: *Set the query string for the health check probes.* *A character string that will be sent to the server to request a performance value; as part of a server agent health check, or to simulate a response as part of an acv health check; may include a character (e.g., ' ')*
eqcli-hc-*hcname* > **quit**	: *Discard all pending health check configuration changes and exit to the top level context.*
eqcli-hc-*hcname* > **response** ***response***	: *Set the response string for the acv health check.* *A character string expected as a response to an ACV query string; may include a character (e.g., ' ')*
eqcli-hc-*hcname* > **show**	: *Display the health check details.*
eqcli-hc-*hcname* > **svname** ***server name***	: *Set the server name to be probed. Required if target =svname.*
eqcli-hc-*hcname* > **target** ***health check target*** **{inherit|internal|svname}**	: *Set the target for the health check probes.*
eqcli-hc-*hcname* > **type** ***health check type*** **{acv|l3|tcp|udp|srvagt|vlb}**	: *Set the type for the health check probes. (Required)*
eqcli-hc-*hcname* > **vlb_mgr** ***vlb manager*** (Only valid when health check 'type' = 'vlb'.)	: *Set the VLB Manager for the health check.*
eqcli-hc-*hcname* > **vlb_param** ***vlb_param*** **{vm_cpu|vm_ram}** (Only valid when health check 'type' = 'vlb'.)	: *Set the VLB Parameter for the health check.*

Using Health Check Commands in Health Check Context	
`eqcli-hc-`*`hcname`* `>` **`vlb_uuid`** ***`vlb_uuid`*** `(Only valid when health check 'type' = 'vlb'.)`	**:** *Set the VLB UUID for the target server.*
`eqcli-hc-`*`hcname`* `>` **`vms`**	**:** *List all the virtual machines from the VLB Manager. Valid only for VLB health check.*

Using Health Check Commands in Health Check Instance Context		
`eqcli obj-obn*-hc-hcn*>` **`commit`**	:	*Commit all pending health check instance configuration changes.*
`eqcli obj-obn*-hc-hcn*>` **`context`**	:	*Display the current command context.*
`eqcli obj-obn*-hc-hcn*>` **`exit`**	:	*Commit all pending health check instance configuration changes and exit to the server instance context.*
`eqcli obj-obn*-hc-hcn*>` **`flags [!]disable],[!]optional`**	:	*Set the health check instance flags.* *"optional" - If other health checks are up, a failure of this health check will not mark the object as 'down'.*
For attached ACV Health Checks:		
`eqcli sp-srv*-hc-hcn*>` **`test`** ***`serverinstancename`***	:	*Test the health check on the referenced object. If the object is a server pool, and a server name is specified after the "test" keyword, only that server instance will be tested. If no server name is specified, all server instances will be tested.* *NOTE: Currently test is only supported on TCP/ACV healthchecks attached to server pools and server instances.*

Health Check Flags

A flag may be turned off by prefixing with "!".

disable	Disables the health check.
icmp_relaxed_probe (L3 health check probes only)	Global Enable/Disable ICMP relaxed probing. When enabled, if a server is DOWN, but has not previously been UP, it will be marked UP. Setting of this flag prevents the sudden reporting of servers as being DOWN following an upgrade.
require_response (VLB and Server Agent health check probes only)	When the **require_response** flag is disabled (the default) and no response is received from the server application, then load balancing decisions are made without considering the health check's returned load value. When the **require_ response** flag is enabled and no response is received from the server application, then the server is marked down and no cluster traffic is sent to it.
probe_ssl (ACV, L4 TCP, and Server Agent health check probes only)	If enabled, the TCP and ACV probe exchange between Equalizer and a server instance will be performed over an encrypted SSL connection.

IP Reputation Commands

Note - IP Reputation is not supported on GX platforms. This includes Equalizer E250GX, E350GX, E450GX , and E650GX.

Using IP Reputation Commands in Global Context	
eqcli > **reputation block *category|IP list***	: *Set a category or list of IPs to block.*
eqcli > **reputation disable**	: *Disable IP Reputation processing.*
eqcli > **reputation enable**	: *Enable IP Reputation processing.*
eqcli > **reputation fetch**	: *Fetch the current IRDB*
eqcli > **reputation load**	: *Fetch pkg file and load IRDB from it.*
eqcli > **reputation pass *category|IP list***	: *Set a category or list of IPs to pass.*
eqcli > **show reputation *blacklist|whitelist*** **[blacklist: List all IPs in blacklist.]** **[category : List all the categories.]** **[whitelist: List all IPs in whitelist.]**	: *Display the category or list of IPs in the selected list.*
eqcli > **reputation stats *blacklist | whitelist|category*** **[blacklist: List all IPs in blacklist.]** **[whitelist: List all IPs in whitelist.]** **[category : List all the categories.]**	: *Show stats of all blocked IPs or of passed IPs.*

Interface Commands

The **interface** context commands let you configure and manage Equalizer's front panel interface ports. There is a separate context corresponding to each front panel port. Ports are created automatically by the system and cannot be deleted. To view a summary of the current port configuration and status, enter:

```
eqcli > show interface
```

The name of each port is displayed, along with the port's current autonegotiation, duplex, speed, and link status.

Refer to "Viewing Link Status and Port Settings" on page 112 for additional information about using the Interface commands as well as a description of the statistical information available for each port.

Using Interface Commands in the Global Context

```
eqcli > interface port cmds
eqcli > show interface
eqcli > show interface port
eqcli > interface port
```

Port Context Commands

```
eqcli if-port> autonegotiation {force|full|select}
eqcli if-port> duplex {full|half}
eqcli if-port> speed {10|100|1000}
eqcli if-port> show
eqcli if-port> stats
```

Interface Command Notes

The following statistics can be displayed for a selected port using the **`stats`** command. Select a port on the Equalizer display to display statistics the port. The tables below shows a typical port statistics display.

Transmit Counters	
Packets	The total number of transmitted packets on this interface.
bytes	The total number of bytes transmitted on this interface
multicasts	The total number of good broadcast/multicast (e.g., ARP) packets transmitted by this interface.
errors	The total number of bad packets transmitted by this interface.
collisions	The total number of packets that were dropped (e.g., lack of transmit buffer , collision detection). These packets are not transmitted by the interface.
Receive Counters	
Packets	The total number of packets received on this interface.
bytes	The total number of bytes received on this interface.
multicasts	The total number of good broadcast/multicast (e.g., ARP) packets received on this interface.
errors	The total number of bad packets (e.g., CRC errors,, alignment errors) received on this interface.
drops	The total number of packets that were dropped (e.g., lack of receive buffer, congestion, invalid classification, e.g., tagged frame received on untagged port) by the receiving interface.
unknown protocol	Tot total number of packets received on this interface that used an unknown protocol.

Link Aggregation Commands

Link aggregation is a means by which multiple physical interfaces are combined into a single logical (aggregated) interface, providing increased bandwidth and failover. The following are CLI commands.

Using Link Aggregation Commands in Global Context	
eqcli > **agr *name***	: *Add or modify an AGR or interface instance.*
eqcli > **show agr**	: *Display a list of AGRs or interface instances.*
eqcli > **show agr *name***	: *Display details for the specified AGR or interface instance*

Using Link Aggregation Commands in an Interface Instance Context	
eqcli > **agr-*name*> flags**	
[!lacp]	
eqcli > **agr-*name*>ifi**	: *Change to the command context for the specified interface instance.*

Link Load Balancing Commands

Using Link Load Balancing Commands in the Global Context	
eqcli > **illb-grp*illb-grp name***	: *Change to the illb group name context.*
eqcli > **illb-grp *illb-grp commands***	: *Modify the illb group*
eqcli > **no illb-grp *illb-grp name***	: *Delete illb-grp name*
eqcli > **show illb-grp]*illb-grp name***	: *Display all illb groups or illb-grp name*
eqcli > **ollb-grp*llb-grp name***	: *Change to the illb group name context.*
eqcli > **ollb-grp *ollb-grp commands***	: *Modify the ollb group*
eqcli > **no ollb-grp *ollb-grp name***	: *Delete ollb-grp name*
eqcli > **show ollb-grp]*ollb-grp name***	: *Display all ollb groups or ollb-grp name*
eqcli > **llb-gw*llb-gw name***	: *Change to the llb gateway name context.*
eqcli > **llb-gw *llb-gw commands***	: *Modify the llb gateway*
eqcli > **no llb-gw *llb-gw name***	: *Delete llb-gw name*
eqcli > **show llb-gw]*llb-gw name***	: *Display all llb gateways or llb-gw name*

LLB Specific Context Commands	
Inbound LLB Group Context Commands	
eqcli > illb-grp-*illbgrpname**> **flags {enable|disable}**	: *Set illb group flags.*
eqcli > illb-grp-*illbgrpname**> **fqdn *fully qualified domain name***	: *Set the illb group Full Qualified Domain Name. Required.*
eqcli > illb-grp-*illbgrpname**> **policy*poltype***	*Set the illb group policy.*
eqcli > illb-grp-*illbgrpname**> **show**	: *Display the illb group configuration.*
eqcli > illb-grp-*illbgrpname**> **target**	: *Change to the command context for the specified illb target.*
eqcli > illb-grp-*name**> **ttl**	: *Set the illb group TTL.*
Outbound LLB Group Context Commands	
eqcli > ollb-grp-*ollbgrpname* > **flags {enable|disable}**	: *Set ollb group flags.*
eqcli > ollb-grp-*ollbgrpname* > **g*wips***	: *Set the ollb group gateway(s). This is a comma-delimited list of LLB gateway IPs.*
eqcli > ollb-grp-*ollbgrpname* > **no**	: *Reset a ollb group parameter to its default value.*
eqcli > ollb-grp-*ollbgrpname* > **show**	: *Display the ollb group configuration.*
LLB Gateway Context Commands	
eqcli > llb-gw-*gwname* > **flags {enable|disable}**	: *Set llb gateway flags.*
eqcli > llb-gw-*gwname* > **gw-hc*objects***	: *Set the llb gateway health_ check(s). This is a comma-delimited list of health_check objects.*
eqcli > llb-gw-*gwname* > **no**	: *Reset an llb gateway parameter to its default value.*
eqcli > llb-gw-*gwname* > **show**	: *Display the llb gateway configuration.*
eqcli > llb-gw-*gwname* > **weight*value***	: *Set llb gateway weight.*

Object List Commands

Object lists make it easier to manage user permissions by allowing an administrator to assign user permissions via list of objects.

An entry in an object list is an “object type” and “object name” pair. Once an object list is created, object list names are used as arguments to user context commands (see "User Commands" on page 222) to give a user permission to access objects in the list.

Using Object List Commands in the Global Context	
`eqcli >` **`objlist`** ***`olname`***	*`: Create an object list, or if it exists change context`*
`eqcli >` **`objlist`** ***`olname`*** **`cmds`**	*`: Modify an object list (see below for cmds)`*
`eqcli >` **`no objlist`** ***`olname`*** **`[`*`force`*`]`**	*`: Delete an object list`*
`eqcli >` **`show objlist [`*`olname`*`]`**	*`: Display all object lists, or the one specified`*

Object List Context Commands	
`eqcli obj-olname>` ***`type object`***	*`: Remove the specified object`*
`eqcli obj-olname>` **`no type`** ***`object`***	*`: Add an object to the list`*
`eqcli obj-olname>` **`show`**	*`: Display object list`*

Object List Notes

- Only a user with the **admin** flag enabled can create, modify, or delete object lists.
- The **type** argument must be one of the following object types: **cert**, **cluster**, **crl**, **geocluster**, **geosite**, **port**, **responder**, **server**, **srvpool**, **subnet**, or **vlan**.
- The **object** argument must be the name of an existing object of the specified **type**. (Object list names and the keyword **all** are not allowed.)
- The **no** form of the **objlist** command is immediately executed; no **commit** is required.

Specifying an Object List When Creating or Modifying an Object

An **objlist** argument is optional when creating (or modifying) an Equalizer object, and adds an entry for the object to the specified object list. To add an entry to an object list, the user must have permission to create objects of the specified type in that object list.

Permission to create objects in an object list is given by the `permit_objlist` command, as outlined in "User Permissions" on page 227.

read and **write** permissions on both the object list and the object to be added to the list (or have the **admin** flag set on the user definition).

Note - When a user creates an object, that user is given **read**, **write**, and **delete** permissions on that object.

Peer Commands

Peer context commands are used to manage the configuration of failover peers, including the failover peer configuration for this Equalizer, which is created when the system is booted for the first time. The default peer name for the Equalizer you are logged into is of the form:

eq_*sysid*

The sysid above is Equalizer's "**Peer sysid**" (or system ID), as displayed in the peer configuration; for example:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: 53
------------------------------------
Peer Name                                            Type    Flags   F/O Mode
         Message(s)
eq_880A4D0B5C94A611CB927D44BED75F20C7BE7D8C (Local)  OS/10   xfr
Standalone-Owner  None

Flags Key:
F/O => failover
A/A => active-active
P/P => preferred_primary
xfr => fo_config_xfer
ssl => use_ssl

eqcli > show peer eq_880A4D0B5C94A611CB927D44BED75F20C7BE7D8C
Peer Name             : eq_880A4D0B5C94A611CB927D44BED75F20C7BE7D8C
Peer signature        :
1RBC880A4D0B5C94A611CB927D44BED75F20C7BE7D8CAC100602
Peer sysid            : 880A4D0B5C94A611CB927D44BED75F20C7BE7D8C
Receive Timeout       : 2
Connect Timeout       : 1
Probe Interval        : 2
Retry Interval        : 5
Strike Count          : 3
Flags                 : fo_config_xfer, local
OS/8 Internal IP      :
Number of Interfaces  : 2
Member of Failover Group        : No
Failover Enabled/Disabled       : Disabled (No remote Peers)
Local/Remote Peer               : Local
Interface                     : v2
State                       : Configure
Substate                    : Object Unchanged
Subnet                      : sn172
State                     : Configure
Substate                  : Object Unchanged
Interface                     : v3
State                       : Configure
```

```
Substate                    : Object Unchanged
Subnet                      : sn192
State                     : Configure
Substate                  : Object Unchanged

eqcli >
```

Using Peer Commands in the Global Context	
eqcli > **peer *peername [cmds]***	: *Create peer (see below for* **cmds***)*
eqcli > **peer *peername cmds***	: *Modify peer (see below for* **cmds**
eqcli > **no peer *peername* [*force***	: *Delete peer*
eqcli > **show peer [*peername*]**	: *Display all peers or a specific peer*
eqcli > **peer *peername***	: *Change to a peer-specific context*

Peer Context Commands	
eqcli > **conn_retry_count *value***	: *Set the Failover connectivity retry count. This controls the number of times that a peer will try to re-gain connectivity with a remote peer before becoming the primary peer for any failover groups running on the remote peer. The default value is 8.*
eqcli peer-peer> **conn_timeout *value***	: *Set the Failover connect timeout (sec)*
eqcli peer-peer> **debug**	: *Set the debug level*
eqcli peer-peer> flags **[!]failover|fo_config_xfer** **[!]os8|[!]preferred_primary** **[!]active-active [!]ssl**	: *Set peer flags (see below)*
eqcli peer-peer> **hb_interval *value***	: *Set the Failover heartbeat interval (seconds).*
eqcli peer-peer> **ipstate**	: *Only valid for local Peer. Displays peer IP states.*
eqcli peer-peer> **os8_intip *internal ip***	: *Set the 'Internal' IP address for an appliance running Version 8 with two VLANs.*
eqcli peer-peer> **name *peer name***	: *Set the peer name.*
eqcli peer-peer> **recv_timeout *value***	: *Set the Failover receive timeout (sec)*
eqcli peer-peer> **retry_interval *value***	: *Set the Failover retry interval (sec)*
eqcli peer-peer> **show**	: *Display the peer configuration.*
eqcli peer-peer> **signature *signature***	: *Set the peer signature.*
eqcli peer-peer> **stats**	: *Display object list*
eqcli peer-peer> **strike_count *value***	: *Set the Failover strike count*

Peer Context Command Flags

A flag may be turned off by prefixing with "!".

`failover`	Adds peer to failover group
`fo_config_xfer`	Enable config transfer between peers
`os8`	Defines peer as OS8 peer
`Preferred_primary`	Sets peer as preferred primary
`active-active`	Enable active/active failover mode

See "Understanding Failover" on page 684 for a complete failover setup procedure.

Remote Management Commands

Remote Management commands are used to specify cipher suites, certificates, and the allowable protocols to use for connection with HTTPS clusters.

Refer to "Using Certificates in HTTPS Clusters" on page 906 for additional information and descriptions on using these commands.

Remote Management Commands in Global Context	
eqcli > **show remote-mgmt**	: *Display the remote management options.*
eqcli > **no remote-mgmt**	: *Delete a specifc remote management option.*
eqcli > **remote-mgmt certificate *certificatename***	: *Set the https server certificate to use.*
eqcli > **remote-mgmt cipherspec *cipherspec***	: *Set the cipherspec to use.*
eqcli > **remote-mgmt protocol *protocol*** **[!]sslv3,[!]tls10,[!]tls11,** **[!]tls12**	: *Set the allowed SSL/TLS protocols.* *NOTE:* *The protocol specification must be enclosed by double quotes if there are any spaces.*

Remote Management Commands in Remote Management Context	
eqcli rm> **show**	: *Display the remote management options.*
eqcli rm> **no *remotemgmtoption***	: *Delete a specific remote management option.*
eqcli rm> **certificate *certificatename***	: *Set the https server certificate to use.*
eqcli rm> **protocol *protocol*** **[!]sslv3,[!]tls10,[!]tls11,** **[!]tls12**	: *Set the allowed SSL/TLS protocols.* *NOTE:* *The protocol specification must be enclosed by double quotes if there are any spaces.*

Responder Commands

Note -Responders are not supported on E250GX model Equalizers.

Responders are global objects in the sense that a single responder can be assigned to multiple clusters. They are used when no servers in the associated server pool are available:

- A responder can be added in the cluster context, in which case it is used when no servers in the server pool defined for the cluster are available.
- A responder can also be assigned to a cluster in a match rule context, in which case the responder is used when no servers in the server pool defined for the match rule are available.

Using Responder Commands in the Global Context	
eqcli > **resp rname *req_cmds***	: *Create **rname** (**req_cmds** = * commands below)*
eqcli > **resp rname *cmd* ...**	: *Modify **rname** (**cmds** = any commands below)*
eqcli > **no resp *rname***	: *Delete **rname***
eqcli > **show resp [*rname*]**	: *Display all responders or **rname***
eqcli > **resp *rname***	: *Change to the "rsp-rname" context (see below)*

Using Responder Commands in a Responder Specific Context	
eqcli rsp-rname> **stats**	: *Display responder statistics*
eqcli rsp-rname> ***type *{sorry|redirect}***	: *(R) MUST SET type FIRST*
type = redirect:	
eqcli rsp-rname> **regex *"expr"***	: *Set redirect regular expression*
eqcli rsp-rname> ***statcode *{301|302|303|307}***	: *Set redirect status code*
eqcli rsp-rname> ***statdesc *"desc"***	: *Set redirect status description*
eqcli rsp-rname> ***url *"url"***	: *Set redirect URL*
type = sorry:	
eqcli rsp-rname> ***html {edit|url}**	: *Set HTML for "sorry" responder*

When creating a responder, you must specify the **type** parameter first on the command line, and then the parameters required for that type. The supported responder types are:

- **redirect** - A standard "HTTP Redirect" response that specifies a return code (**statcode**), description (**statdesc**), and redirect URL (**url**). When the client receives this page, it is automatically redirected to the redirect URL. Redirect pages can be configured to use parts of the request URL in the HTTP Redirect response (using an optional regular expression).
- **sorry** - A customized HTML "sorry page" that can, for example, ask the client to retry later or go to another URL

For example, the following command creates a **sorry** responder named **Sorry01**, and downloads the redirect URL from the URL specified on the command line:

```
eqcli > resp Sorry01 type sorry html ftp://mylocalftpserver/redirect.html
```

The contents of the file *redirect.html* will be used as the redirect URL for the responder.

The **html** parameter can be specified on the command line as follows:

html	`edit`	Launch an editor to supply the HTML for the sorry page.
	`"url"`	Download the redirect URL from the ftp:// or http:// protocol URL supplied on the command line (quotes are optional).

Regular Expressions in Redirect Responders

For a discussion of regular expressions and how they can be used in redirect type responders, see "How to Use Regular Expressions" on page 845

Server Commands

In the server context, you define a real server using a minimal set of parameters (IP address, port, protocol, etc.). Once defined, a real server can then be associated with one or more server pools, which in turn are associated with one or more Layer 4 clusters or Layer 7 match rules.

Using Server Commands in the Global Context	
`eqcli > server svname req_cmds`	: *Create **svname** (**req_cmds** = * commands below)*
`eqcli > server svname cmds`	: *Modify **svname** (**cmds** = any commands below)*
`eqcli > no server svname`	: *Delete **svname***
`eqcli > show server [svname]`	: *Display all servers or **svname***
`eqcli > server svname`	: *Change to the "sv-svname" context(see below)*

Using Server Commands in a Server Specific Context	
`eqcli sv-svname> hc health check name`	: *Change to the command context for the specified health check instance.*
`eqcli sv-svname> *ip ip_addr`	: *Server IP address*
`eqcli sv-svname> no {max_reuse_conn|reuse_conn_to}`	: *Reset the parameter to its default value*
`eqcli sv-svname> *proto {tcp|udp}`	: *Server protocol*
`eqcli sv-svname> *port integer`	: *Server port*
`eqcli sv-svname> show`	: *Show server configuration*
`eqcli sv-svname> stats`	: *Display server statistics*
`eqcli sv-svname> max_reuse_conn integer`	: *Maximum number of connections to this server*
`eqcli sv-svname> reuse_conn_to integer`	: *Timeout for connection re-use*
`eqcli sv-svanme> uuiduuidname`	: *Associate a virtual machine with the server.*
`eqcli sv-svanme> vlb_manager vlbmgrname`	: *Attach a VLB Manager for the associated virtual machine.*
`eqcli sv-svanme> vms`	: *List all the virtual machines from the VLB Manager.*

The **`max_reuse_conn`** and **`reuse_conn_to`** are used to set operating parameters for HTTP multiplexing. HTTP multiplexing is disabled by default, and is turned on/off using the **`tcp_mux`** cluster flag. Refer to "HTTP Multiplexing" on page 768 for additional information.

Server Pool, Server Instance, and Caching Commands

A server is attached to a cluster via a *server pool*. A server pool is a collection of server definitions, each of which has additional parameters assigned to it in the server pool -- these additional parameters are organized by the server's name and are referred to as *server instances* within the server pool context. This allows you to associated a distinct set of server instance options (weight, flags, maximum number of connections), to multiple instances of the same real server in different server pools.

Using Server Pool Commands in the Global Context		
eqcli > **srvpool spname commit**	:	*Commit all pending server pool configuration changes and do not change the command context.*
eqcli > **srvpool spname context**	:	*Displays the current command context.*
eqcli > **srvpool spname custom_actconn *0-100***	:	*Set the active connections weight for the custom load balancing policy.*
eqcli > **srvpool spname custom_delay *0-100***	:	*Set the delay weight for the custom load balancing policy.*
eqcli > **srvpool spname flags** flag **{[!]disable}**	:	*Set the server pool flags.*
eqcli > **srvpool spname policy *round-robin|static|adaptive|response|least-cxns|server-agent***	:	*Set the load balancing policy for a server pool. Required. (see below)*
eqcli > **srvpool spname quit**	:	*Discard all pending server pool configuration changes and exit to the global context.*
eqcli > **srvpool spname respv *load balancing responsiveness***	:	*Set the load balancing responsiveness for a server pool. Required.*
eqcli > **srvpool spname si *siname***	:	*Change to the command context for the specified server instance.*
eqcli > **srvpool spname stats**	:	*Display server pool statistics.*

Using Server Pool Commands in a Server Pool Specific Context	
`eqcli sp-`*`spname`*`>` **`custom_actconn`** ***`percent`***	*`: Custom LB policy - active connections percentage`*
`eqcli sp-`*`spname`*`>` **`custom_delay`** ***`percent`***	*`: Custom LB policy - server delay percentage`*
`eqcli sp-`*`spname`*`>` **`no {si|srvpool|}`** ***`name parameter`***	*`: Delete a server instance or reset a server pool or server instance parameter.`*
`eqcli sp-`*`spname`*`>` **`policy`** ***`policyname`***	*`: Set the load balancing policy for a server pool. Required.`*
`eqcli sp-`*`spname`*`>` **`quit`**	*`: Discard all pending server pool configuration changes and exit to the global context.`*
`eqcli sp-`*`spname`*`>` **`respv`** ***`load balancing responsiveness`***	*`: Set the load balancing responsiveness for a server pool. Required.`*
`eqcli sp-`*`spname`*`>` **`show si`**	*`: Display the current server pool configuration, list all server instances or display details for the specified server instance`*
`eqcli sp-`*`spname`*`>` **`si`** ***`server instance name`***	*`: Change to the command context for the specified server instance.`*
`eqcli sp-`*`spname`*`>` **`stats`**	*`: Display server pool statistics.`*

Using Caching Commands Server Pool Specific Context	
eqcli sp-*spname*-cache> **age *value***	: *Set the age (in seconds)at which an item in the cache is marked 'stale', and is therefore a candidate for removal or replacement. If the cached object reaches this age limit, it becomes 'stale'.(Default: 24 hours/86400 seconds)*
eqcli sp-*spname*-cache> **flags** **{[!]disable}**	: *Set the cache flags* *disable := Disable the cache (all existing cache will be flushed)* *enable := Enable the cache* *(Default: enable)*
eqcli sp-*spname*-cache> **max_object_ size *integer***	: *Set the maximum object size (in MB)to cache. If the response is larger than the value set, it will not be cached.* *Minimum: 0.* *Maximum: The maximum cache size is platform-dependent as follows:* *E370LX -- 100MB* *E470LX -- 200MB* *E670LX -- 300MB* *E970LX -- 500MB* *EQOD -- 50MB*
eqcli sp-*spname*-cache> **max_ size*integer***	: *Set the maximum amount of memory allocated, in bytes, to the server pool cache* *Minimum: 0* *Maximum: 10MB max (10485760 bytes)* *(Default: 2KB (2048 bytes)*
eqcli sp-*spname*-cache> **policy *policy***	: *Set the replacement policy for the cache.* *policy := oldest, largest* *oldest := the entry with the oldest freshness time will be removed.* *largest := the largest item will be removed from the cache.* *(Default: oldest)*
eqcli sp-*spname*-cache> **show**	: *Display all cache parameters*
eqcli sp-*spname*-cache> **stats**	: *Display all of cache statistics on this server pool.*

Using Server Instance Commands in the Server Instance Context

```
eqcli sp-srv*-si-siname > flags flag                   : Set the server instance
   {[!]hot_spare, [!]persist_override,                   flags.(see below)
    [!]quiesce, [!]strict_maxconn}

eqcli sp-srv*-si-siname > maxconn maxconn              : Set the number of maximum
                                                         connections for a server
                                                         instance.

eqcli sp-srv*-si-siname > stats                        : Display server instance
                                                         statistics.

eqcli sp-srv*-si-siname > weight value                   Set the server instance
                                                         initial weight. Required.
```

Server Instance Flags

A flag may be turned off by prefixing with "!".

Flag	Description
hot_spare	Enable the hot spare check box if you plan to use this server as a backup server, in case the other server instances in a server pool on the cluster fail. Enabling hot spare forces Equalizer to direct incoming connections to this server only if all the other servers in the cluster are down. You should only configure one server in a cluster as a hot spare.**persist_override** - Disables persistence for the server when the persist flag (Layer 7 cluster) or a non-zero sticky time (Layer 4 cluster) is set on a cluster. For a Layer 7 cluster, this means that a cookie will not be inserted into the response header when returned to the client. No sticky record is set for a Layer 4 cluster. This flag is usually used to disable persistence for a hot spare.
persist_override	When enabled, Layer 4 sticky records and Layer 7 cookies are ignored for the server instance.
quiesce	When enabled, Equalizer avoids sending new requests to the server. This is usually used in preparation for shutting down an HTTP or HTTPS server, and is sometimes also called “server draining”.

Flag	Description
`strict_maxconn`	This flag allows you to customize the behavior of the max connections parameter (see above). When **Strict Max Cx** is enabled (the default), the max connections parameter is interpreted as a strict maximum and is never overridden. If a client attempts to connect to a server that has a number of connections equal to the max connections setting, then the connection is refused. When **Strict Max Cx** is disabled, the max connections setting will be overridden in any of the following circumstances: A client attempts to connect to a server with the hot spare flag enabled - this allows hot spares to service more than the max connections setting of connections. A client attempting to connect to a Layer 7 cluster has a persistence cookie and the server identified in the cookie has already reached its max connections limit. A client attempting to connect to a Layer 4 cluster has an existing sticky persistence connection to a server and that server has already reached its max connections limit.

Load Balancing Policies

Equalizer supports the following load balancing policies, each of which is associated with a particular algorithm that Equalizer uses to determine how to distribute requests on a server pool in the cluster:

- **Round-robin** load balancing - distributes requests equally on the server pool in the cluster. Equalizer dispatches the first incoming request to the first server, the second to the second server, and so on. When Equalizer reaches the last server, it repeats the cycle. If a server in the cluster is down, Equalizer does not send requests to that server. This is the default method.

  The round robin method does not support Equalizer’s adaptive load balancing feature; so, Equalizer ignores the servers’ initial weights and does not attempt to dynamically adjust server weights based on server performance.
- **Static** load balancing - distributes requests among the servers depending on their assigned initial weights. A server with a higher initial weight gets a higher percentage of the incoming requests. Think of this method as a *weighted round robin* implementation. Static weight load balancing does not support Equalizer’s adaptive load balancing feature; Equalizer does not dynamically adjust server weights based on server performance.
- **Adaptive** load balancing - distributes the load according to the following performance indicators for each server.
- **Response** is the length of time for the server to begin sending reply packets after Equalizer sends a request.
- **Least Cxns** (least connections) load balancing - dispatches the highest percentage of requests to the server with the least number of active connections. In the same way as Fastest Response, Equalizer tries to avoid overloading the server so it checks the server’s response time and server agent value. Least Connections optimizes the balance of connections to servers in the cluster.
- **Server Agent** load balancing - dispatches the highest percentage of requests to the server with the lowest server agent value. In a similar way to Fastest Response, Equalizer tries to avoid overloading the server by checking the number of connections and response time. This method only works if the server agents are running on every server instance in the server pool.
- **Custom** load balancing - If custom is selected, you can adjust the load balancing policy parameters:
  - **Connections** - The relative influence on the policy of the number of active connections currently open to a server

- **Response Time**load balancing -dispatches the highest percentage of requests to the server with the shortest response time. Equalizer does this carefully: if Equalizer sends too many requests to a server, the result can be an overloaded server with slower response time. The fastest response policy optimizes the cluster-wide response time. The fastest response policy also checks the number of active connections and server agent values (if configured); but both of these have less of an influence than they do under the adaptive load balancing policy. For example, if a server's active connection count and server agent values are high, Equalizer might not dispatch new requests to that server even if that server's response time is the fastest in the cluster.
- **Responsiveness** - Determines how aggressively server dynamic weights are optimized as a cluster load changes.

Server Side Encryption Commands

Using Server Side Encryption Commands in Global Context	
`eqcli >` **`show sse`**	*`: Display the sse configuration.`*
`eqcli >` **`sse cipherspec`** ***`cipherstring`***	*`: Set the sse cipherspec`*
`eqcli >` **`no sse`**	*`: Reset one or more sse parameters.`*
`eqcli >`**`sse flags {[!]allow_tls10, [!]allow_tls11, [!]allow_tls12}`**	*`: Set the sse flags`*

Smart Control Commands

The **smart_control** commands let you configure and manage Smart Controls.

Refer to "Adding Smart Events" on page 799 for a description of the process for adding Smart Controls.

To view a summary of the currently configured Smart Controls:

```
eqcli > show smart_control smart_control_name
```

The names of all of the currently configured Smart Controls will be displayed.

Using Smart Control Commands in the Global Context

```
eqcli > smart_controls name
eqcli > show smart_controls name
eqcli > no smart_controls name
eqcli > show smart_controls
```

Smart Control Context Commands	
eqcli *sc-scname* > **flags** **{[!]disable}**	: *Set Smart Control flags.*
eqcli *sc-scname* > **interval** ***seconds***	: *Set the Smart Control interval.*
eqcli *sc-scname* > **lastrun**	: *Display the first 1023 characters of the output generated during the last execution of this smart control.*
eqcli *sc-scname* > **run_limit** ***seconds***	: *Set the Smart Control run limit. (default: 10 seconds)*
eqcli *sc-scname* > **run**	: *Run Smart Control now.*
eqcli *sc-scname* > **schedule*schedule string***	: *Set the Smart Control schedule.* *The string is in the standard cron format, but with an additional first column -- second:* *second 0-59* *minute 0-59* *hour 0-23* *day of month 1-31* *month 1-12 (or names, see below)* *day of week 0-7 (0 or 7 is Sun, or use names)* *Lists and ranges are supported (use ',' as a list delimiter or '-' as a range delimiter), but steps are not.* *White space (' ' or) is a column break, and consecutive white spaces are treated as one.* *Fields which are an asterisk ('*') are skipped.* *Day names: 'Mon','Tue','Wed','Thu','Fri','Sat','Sun' Month names: 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'* *Note: The schedule string must be enclosed in quotes.* *i.e.: "* 0,30 * * * Mon" would be translated as 'Every Monday, run this every 30 seconds'.*
eqcli *sc-scname* > **script *mode*** **edit|URL**	: *edit invokes an editor to enter the desired script.* *<URL> fetches the script from the entered fully qualified ftp/http site.*
eqcli *sc-scname* > **stats**	: *Display Smart Control statistics.*

SNMP Commands

The parameters in the SNMP context specify return values for the following Object IDs (OIDs) in the Equalizer SNMP Management Information Base (MIB):

OID	Parameter	Default Value	Description
	community	Equalizer	Any SNMP management console needs to send the correct community string along with all SNMP requests. If the sent community string is not correct, Equalizer discards the request and will not respond.
	contact	public	Contact is the name of the person responsible for this unit.
	description	Equalizer	This is the user-assigned description of the Equalizer.
	location	location	Location describes Equalizer's physical location.
	name	Equalizer	This is the name assigned to the system. By default it is Equalizer .
	server	VLAN IP	To configure the SNMP agent to listen and respond on a particular IP address, enter the address.
	serverport (optional)	162	This is optional. If not entered, the default trap server port (162) will be used.

The following tables list the SNMP context commands:

Using SNMP Commands in the Global Context

```
eqcli > no snmp cmd                : Reset the specified parameter to its default
                                     value

eqcli > show snmp                  : Display SNMP parameter settings

eqcli > snmp                       : Change to the "snmp" context (see below)
```

SNMP Context Commands		
eqcli snmp> **community *string***	:	*Set the SNMP community address.*
eqcli snmp> **contact *string***	:	*Set the SNMP contact address.*
eqcli snmp> **description *string***	:	*Set the SNMP description*
eqcli snmp> **location *string***	:	*Set the SNMP location.*
eqcli snmp> **name *string***	:	*Set the SNMP name.*
eqcli snmp> **server server list**	:	*A single trap server or a list of trap servers in <IP:port> format. Examples:* *server 10.0.0.121:80* *server 10.0.0.121,10.0.0.120:162* *Note: If 'port' is not specified, it defaults to 162.*
eqcli snmp> **show**	:	*Display SNMP parameter configuration*

Downloading Equalizer MIB Files

The MIB files can be downloaded from Equalizer using a browser pointed at:

```
http://<Equalizer>/eqmanual/<mibname>.my
```

Tunnel Commands

Use tunnel context commands to configure Equalizer to access the IPv6 Internet via an IPv6 “6in4” tunnel. Note that you must first request a tunnel configuration from a tunnel broker before setting up the tunnel endpoint on Equalizer. See "IPv6 Tunnel Overview" on page 316 for more information.

Using Tunnel Commands in the Global Context	
eqcli > **tunnel *tname* *[cmds]***	: *Create tunnel* ***tname*** *(see below for* ***cmds****)*
eqcli > **tunnel *tname* *cmds***	: *Modify tunnel* ***tname*** *(see below for cmds)*
eqcli > **no tunnel *tname***	: *Delete tunnel* ***tname***
eqcli > **show tunnel[*tname*]**	: *Display all tunnels or a specific tunnel*
eqcli > **tunnel *tname***	: *Change to a tunnel context (see below)*

Tunnel Context Commands	
eqcli tl-*tname*> ***local_address *ipv6_addr***	: *Local IPv6 address from broker*
eqcli tl-*tname*> ***local_endpoint *ipv4_addr***	: *Local IPv4 address*
eqcli tl-*tname*> ***remote_address *ipv6_addr***	: *Remote IPv6 address from broker*
eqcli tl-*tname*> ***remot_endpoint *ipv4_addr***	: *Remote IPv4 address from broker*
eqcli tl-*tname*> **show**	: *Display tunnel settings*
eqcli tl-*tname*> ***type *ipip***	: *Tunnel type (only ipip supported)*

User Commands

Using "User" Comands in the Global Context	
`eqcli > user uname [cmds]`	: *Create user **uname** (see below for **cmds**)*
`eqcli > user uname cmds`	: *Modify user **uname** (see below for cmds)*
`eqcli > no user uname`	: *Delete user **uname***
`eqcli > show user [uname]`	: *Display all users or a specific user*
`eqcli > user uname`	: *Change to the "user-login" context (see below)*

"User" Context Comands	
`eqcli user-uname > alert alert name`	: *Set a user alert.*
`eqcli user-uname > duration seconds`	: *Set the idle login timeout*
`eqcli user-uname > flags {[!]admin, [!]read_global, [!]write_global [!]primary}`	: *Permission Flags:* *admin = read/write all objects & parameters* *read_global = read global parameters* *write_global = modify global parameters* *Failover Flag:* *primary = When in failover, controls whether the system will generate alerts only for failover groups that are in primary mode on the system, or for all failover groups. (See "User Flags" on page 226 for additional details)*
`eqcli user-uname > no duration`	: *Set default duration (0)*
`eqcli user-uname > no permit_object perm type object`	: *Remove permission on object*
`eqcli user-uname > no permit_objlist perm type objlist`	: *Remove perm from objlist*
`eqcli user-uname > password`	: *Change user password*
`eqcli user-uname > permit_object perm type object`	: *Add permission on object*
`eqcli user-uname > locale {[!]en,`	: *To set English locale*
`[!]ja}`	: *To set Japanese locale*

"User" Context Comands
eqcli user-*uname* > **show** : *Display user settings*

User-Alert Context Commands		
eqcli > user-*uname*-*alertname* > **alert_type *alert flags*** **[!] exception [!] state_change**	:	*Set the alert type. Required.* *The alert flags specification must be enclosed by double quotes if there are any spaces.*
eqcli > user-*uname*-*alertname* > ***from email-address***	:	*Set the from email address.*
eqcli > user-*uname*-*alertname* > **no *smart_ control***	:	*Delete the specified smart control name(s)*
eqcli > user-*uname*-*alertname* > **notify_ type notify *notify flags*** **[!]email,[!]ui,[!]snmp,[!]syslog, [!]smartd**	:	*Set the alert notify flags. Required. The notify flags may be delimited by ',' or '|'.* *WARNING: The notify flags specification must be enclosed by double quotes if there are any spaces.*
eqcli > user-*uname*-*alertname* > **object *fully-qualified object name***	:	*Set the object fully-qualified object name. Required. The object name is the name of an object, existing in the configuration, for which this alert definition is to be applied. The 'fully-qualified object name' is a semi-colon delimited list describing the object hierarchy. For example, to set an alert for vlan vl01, subnet sn00, the user would specify: object vl01:sn00, and the object_type would be 'subnet'.* *For another example, to set an alert for peer eq-A, F/O Group fo_group1, the user would specify:* *object eq-A:fo_group1, and the object_type would be 'fogrp'.* *Note: The last object name in the hierarchy may contain one or more wildcard (*) characters. For example, to configure an alert for all subnets of vlan vl01, specify: 'object vl01:*' Also, an object hierchary of 'vl01:sn*1' would configure alerts for subnets: sn1, sn01, sn11, etc.*

User-Alert Context Commands		
eqcli > user-*uname-alertname* > **object_ type *object-type***	:	*Set the object type. Required.* *Object type can be server, cluster, match, srvpool, si, resp, peer, vlan, subnet, geocluster, geosite, gsi, interface, user, certificate, crl, route, tunel, license, health_check, hci, vlb_manager, resource, ri, external_ services, smtp_relay,* *fogrp*
eqcli > user-*uname-alertname* > quit	:	*Discard all pending alert configuration changes and exit to the user context.*
eqcli > user-*uname-alertname* > show	:	*Display the alert details.*
eqcli > user-*uname-alertname* > **smart_ control *name***	:	*Add one or more smart_control objects to the alert where 'name' is the name of a single smart_control name, or a comma-separated list of smart_control names*
eqcli > user-*uname-alertname* > state	:	*Set the alert state flag. enable/disable*
eqcli > user-*uname-alertname* > **subject *user string***	:	*Set the subject.* *User string is any (up to 256) characters the user wishes to* *enter. It must be surrounded by quotes if it has embedded blanks. Its usage depends upon the notify_type.*
eqcli > user-*uname-alertname* > to ***to email addresses***	:	*Set the email address(es).* *Email addresses are email1,email2,...emailx, where:* ***email= user@<domain***

User Alert Notify Type Flags

A flag may be turned off by prefixing with "!".

`email`	When enabled, sends an email to the specified recipients, using a specified SMTP relay mail server. When this notification type is used, an email address is also required. A subject line for the email is optional
`ui`	When enabled this notifies users of an alert in the CLI.
`snmp`	When enabled, allows SNMP traps to enable an agent to notify a management station of significant events by way of unsolicited SNMP messages.
`syslog`	When enabled, sends an alert message to the system log

User Flags

User flags are used to override permissions checks, as follows:

Flag	Description
`admin`	All permissions checks are overridden for the user (including **`read_global`** and **`write_global`**). The user has complete administrative control over the system. Only users with the admin flag can: **`read`**, **`write`**, and **`delete`** any object on which they do not have explicit permission **`write`**, **`create`**, and **`delete`** object lists and user definitions (with the exception of a user changing their own password) **`read_global`** - User can do a show in the global context.
`write_global`	User can modify global parameters and execute global commands other than show in the global context.
`read_global`	User can read all global parameters.
`primary`	When in failover, controls whether the system will generate alerts only for failover groups that are in primary mode on the system, or for all failover groups. If enabled, alerts for load balancing objects will only be generated for the associated failover groups that are in "Primary" mode. If disabled, alerts for load balancing objects will be generated for all failover groups,regardless of mode. If not in failover, this flag has no effect.

Setting the Locale

You can set the locale for Equalizer to either English or Japanese (2 available options at this time). The default locale is “en” for English.

For English enter the following:

```
eqcli user-uname> locale en
```

For Japanese enter the following:

```
eqcli user-uname> locale ja
```

Creating a User

When a user name is created:

- A default user (i.e. "touch") is assigned a **`duration`** of 0 seconds . When additional users are created the default **`duration`** value is 3600 seconds.
- The user creating the new user name is prompted for a password (regardless of whether they specified the **`password`** keyword on the command line) .

Deleting a User

The **`no user`** command is immediately executed and the user name is removed, with one exception: if the user name is the only one with the **`admin`** flag enabled, the user name is not removed.

User Passwords

The **`password`** command allows a logged in user to change the password for their user name. A user name with the **`admin`** flag can modify the password for any user name. The password itself is not permitted on the command line, and is not displayed by a user context **`show`** command (or any eqcli command).

User Permissions

When a user attempts to access an object (cluster, server, server pool, VLAN, etc.) on Equalizer, the system determines whether the user has permission to access the object as follows:

1. If the user's definition has the **`admin`** flag enabled, then access is granted.
2. Otherwise, the user must have specific permission granted on the object for the access mode being attempted. For example, if the user attempts to display a cluster, then the user must have **`read`** permission on the cluster.

Permission to access an object is granted in one of two ways:

- The **`permit_object`** command gives the user the specified access permissions on the specified object.
- The **`permit_objlist`** command gives the user access permissions on all objects of a particular type as listed in the object list specified on the command line.

Note - The **`permit_object`** and **`permit_objlist`** commands:

- can be used only on existing user logins.

- must be entered one at a time, on a line by themselves, with no other user context commands on the command line

So, for example, you cannot modify a user's **`duration`** parameter and in the same command line include a **`permit_object`** or **`permit_objlist`** command.

Using permit_object to Assign User Permissions on a Single Object

The **user** context **permit_object** command has the following syntax:

permit_object ***perm type object_name***

The command assigns the given permission on the given object in the user context. The command arguments are as follows:

- **perm -** One or more of the following permissions: **read**, **write**, **delete**. Multiple permissions must be separated by commas. If spaces are included, the entire list of permissions must be enclosed in quotes.
- **type -** One of the following object types: **cert**,**cluster**,**crl**,**geocluster**,**geosite**,**port**,**server**,**srvpool**,**subnet**,**user**,**vlan**.
- **object_name -** The name of an existing object of the *type* given on the command line.

For example, the following command executed in the global context assigns **read** and **write** permission to the server **sv00** for the existing login **user1**:

```
eqcli > user user1 permit_object read,write server sv00
```

Using permit_objlist to Assign User Permissions on a Group of Objects

The **user** context **permit_objlist** command has the following syntax for assigning **read**, **write**, and **delete** permissions:

```
permit_objlist perm type objlist_name
```

This form of the **permit_objlist** command assigns the given permission (**perm**) on all objects of the specified **type** that appear in the object list specified by **objlist_name**. The command arguments for assigning permission to objects in an object list are as follows:

- **perm -** One or more of the following permissions: **read**, **write**, **delete**. Multiple permissions must be separated by commas. If spaces are included, the entire list of permissions must be enclosed in quotes.
- **type -** One of the following object types: **cert**,**cluster**, **crl**,**geocluster**,**geosite**,**port**,**server**,**srvpool**,**subnet**,**user**,**vlan**.
- **objlist_name -** The name of an existing object list.

For example, the following command executed in the global context assigns **read** and **write** permission to all of the servers listed in the object list **objlist1** for the login **user1**:

```
eqcli > user user1 permit_objlist read,write server objlist1
```

For more information on object lists, please see "Object List Commands" on page 200.

Using permit_objlist to Allow a User to Create Objects

The **user** context **permit_objlist** command has the following syntax for assigning the **create** permission to a user:

```
permit_objlist create type {default | objlist_name}
```

- This form of the **permit_objlist** command allows the user to **create** objects of the specified **type**. The command arguments for assigning permission to objects in an object list are as follows:
- **type** - One of the following object types: **cert**,**cluster**,**crl**,**geocluster**,**geosite**,**port**,**server**, **srvpool**,**subnet**,**user**,**vlan**.
- **default** - Specifies that objects created by this user will only be visible to the user creating the object and any user with the **admin** flag set.
- **objlist_name** - Specifies that the user can supply the given object list name as an argument when creating objects of the specified **type**. An entry for the created object is placed in the object list. Objects created in this manner will be visible to other users who have permission to use this object list.

For example, the following command executed in the global context allows **user1** to create servers that other non-admin users cannot access:

```
eqcli > user user1 permit_objlist create server default
```

The following command allows **user1** to create servers and specify the **objlist1** object list when creating a server, thus adding the new server to **objlist1**:

```
eqcli > user user1 permit_objlist create server objlist1
```

User Permissions Assigned on Object Creation

When an object is created, the user creating the object is given **read**, **write**, and **delete** permissions for the object.

Displaying User Information

In the **user** context, a **show** command displays the user settings for **duration** and **flags**, followed by the user permission list.

VLAN and Subnet Commands

Using VLAN Commands in the Global Context	
eqcli > **vlan *vlname req_cmds***	: *Create **vlname** (**req_cmds** = * commands below)*
eqcli > **vlan *vlname cmds***	: *Modify **vlname** (**cmds** = any commands below)*
eqcli > **no vlan *vlname***	: *Delete **vlname***
eqcli > **show vlan *[vlname]***	: *Display all VLANs or **vlname***
eqcli > **vlan *vlname***	: *Change to the "vl-vlname" context (see below)*

VLAN Specific Context Commands	
eqcli vl-*vlname*> **ifi**	: *Change to the command context for the specified interface instance.*
eqcli vl-*vlname*> **show**	: *Display VLAN configuration*
eqcli vl-*vlname*> **subnet *subname***	: *Change to subnet specific context*
eqcli vl-*vlname*> **subnet *subname cmd***	: *Execute subnet specific command*
eqcli vl-*vlname*> **mtu** [***mtuvalue***]	: *Set the vlan MTU.*
eqcli vl-*vlname*> ***vid *integer***	: *Set VLAN ID.(Value between 1 and 4094)*

VLAN Commands in the ifi Context	
eqcli **vl-*vlname*-ifi-*ifiname*** > **type**	: *Set the interface instance vlan type <tagged|untagged>*
eqcli **vl-*vlname*-ifi-*ifiname*** > **show**	: *Display the ifi configuration*

Using Subnet Commands in the Global Context	
eqcli > **vlan *vlname* subnet *subname* *req_cmds***	: *Create* **subname (req_cmds** *= * commands below)*
eqcli > **vlan *vlname* subnet *subname* *cmds***	: *Modify* **subname (cmds** *= any commands below)*
eqcli > **no vlan *vlname* subnet *subname***	: *Delete* **subname**
eqcli > **show vlan *vlname* subnet [*subname*]**	: *Display all subnets or* **subname**
eqcli > **vlan *vlname* subnet *subname***	: *Change to a subnet context.*

Subnet Specific Context Commands	
eqcli vl-*vlname*-sn-*subname*> **flags [!]heartbeat}**	: *Set subnet flags*
eqcli vl-*vlname*-sn-*subname*> **force**	: *Force the subnet modification, ignoring any conflicts.*
eqcli vl-*vlname*-sn-*subname*> **from *ip_addr***	: *Set NAT from IP (with or without CIDR notation).*
eqcli vl-*vlname*-sn-*subname*> **hb_interval**	: *Set the heartbeat probe interval for a subnet.*
eqcli vl-*vlname*-sn-*subname*> **ip**	: *Set the subnet IP address. Required.*
eqcli vl-*vlname*-sn-*subname*> **nat**	: *Add or modify a subnet nat.*
eqcli vl-*vlname*-sn-*subname*> **no *parameter***	: *Delete a route or reset a subnet parameter to its default value.*
eqcli vl-*vlname*-sn-*subname*> **no route *src* *dest***	: *Remove a route*
eqcli vl-*vlname*-sn-*subname*> **out*ip_addr***	: *Set NAT out IP.*
eqcli vl-*vlname*-sn-*subname*> **permit**	: *Set the list of permitted subnets on a subnet.*
eqcli vl-*vlname*-sn-*subname*> **route**	: *Add or modify a subnet route.*
eqcli vl-*vlname*-sn-*subname*> **services {!] http,[!]https, [!]ssh, [!]snmp, [!]envoy, [!]envoy_agent, [!]fo_http, [!]fo_https, [!]fo_ssh,[!]fo_snmp, [!]fo_envoy,[!]fo_envoy_agent}**	: *Subnet Services (see below)*
eqcli vl-*vlname*-sn-*subname*> **show**	: *Display subnet*

Subnet Specific Context Commands	
`eqcli vl-`*`vlname`*`-sn-`*`subname`*`>` **`strike_count`** ***`integer`***	: *Set the strike count threshold for a subnet. When the number of strikes detected on this subnet exceeds this value, the subnet has failed. A value of 0 indicates this subnet will never be considered failed.*
`eqcli vl-`*`vlname`*`-sn-`*`subname`*`>` **`virt_addr`** ***`cidr_addr`***	: *Set the subnet Failover IP address.*

VLAN Subnet Flags

A flag may be turned off by prefixing with "!".

Flag	Description
heartbeat	Allows the failover peers to probe one another over the subnet. At least one subnet must have a Heartbeat flag enabled.

VLAN Subnet Services

Services may be turned off by prefixing with "!".

Service	Description
`http`	When enabled, the Equalizer will listen for HTTP connections on Equalizer's IP address on the subnet. The global HTTP GUI service must also be enabled.
`https`	When enabled, the Equalizer will listen for HTTPS connections on Equalizer's IP address on the subnet. The global HTTPS GUI service must also be enabled.
`ssh`	When enabled, SSH login will be permitted on Equalizer's IP address on the subnet. The global SSH service must also be enabled.
`snmp`	When enabled, SNMP will accept connections on Equalizer's IP address on the subnet. The global SNMP service must also be enabled.
`envoy`	When enabled, Envoy will accept DNS lookup connections on Equalizer's IP address on the subnet. The global Envoy service must also be enabled.
`envoy_agent`	When enabled, Envoy health checks will be performed on the subnet using Equalizer's IP address on the subnet as the source IP. The global Envoy Agent service must also be enabled.
`fo_http`	When enabled, the Equalizer will listen for HTTP connections on Equalizer's Failover IP address (if configured) on the subnet. The global HTTP GUI service must also be enabled.
`fo_https`	When enabled, the Equalizer will listen for HTTPS connections on Equalizer's Failover IP address (if configured) on the subnet. The global HTTPS GUI service must also be enabled.
`fo_ssh`	When enabled, SSH login will be permitted on Equalizer's Failover IP address (if configured) on the subnet. The global SSH service must also be enabled.
`fo_snmp`	When enabled, SNMP will accept connections on Equalizer's Failover IP address (if configured) on the subnet. The global SNMP service must also be enabled.
`fo_envoy`	When enabled, Envoy will accept DNS lookup connections on Equalizer's Failover IP address (if configured) on the subnet. The global Envoy service must also be enabled.
`fo_envoy_agent`	When enabled, Envoy health checks will be performed on the subnet using Equalizer's Failover IP address on the subnet as the source IP. The global Envoy Agent service must also be enabled.

VLAN and Subnet Command Notes

The **vlan** context defines Equalizer’s network connectivity. Each VLAN definition defines the front panel ports that are configured for the VLAN, the VLAN ID (VID), and the subnets that belong to the VLAN.

VLAN Subnets

A single VLAN can have more than one subnet assigned to it. In most configurations, there is a one-to-one relationship between VLANs and subnets, but some practical problems are sometimes solved by adding an additional subnet to a VLAN. For example, if all the IP addresses on the subnet assigned to a VLAN are exhausted, the easiest way to add more IP addresses without reconfiguring the network is to add an additional subnet to the VLAN.

VLAN IP Addresses

A VLAN IP address is defined on all subnets in a VLAN and is Equalizer’s IP address on that subnet. Subnet IP addresses must be specified in CIDR format (e.g. 172.16.0.200/21). A VLAN can contain multiple subnets with a mix of IPv4 and IPv6 addresses on different subnets in the same VLAN.

VLAN Services

A VLAN can have several **services** running on it: the GUI can be available on the VLAN IP address via HTTP and/or HTTPS; and, SSH login on the VLAN IP can be enabled as well. It is not required that any of these services be enabled on any VLAN.

If **services** are enabled on the VLAN, they must also be enabled in the global context in order to be functional on the VLAN. See the **services** command in "Global Commands" on page 164.

Routing Between VLANs

By default, packets are not routed between VLANs. In other words, if a packet for a destination address that is configured on **vlan2** arrives at a port that is configured for **vlan1** *only*, the packet is dropped. Routing from **vlan1** to **vlan2** is configured by adding **vlan2** to the list of permitted VLANs for **vlan1**.

For example, let’s say port 1 is configured for **vlan1** and subnet 10.10.10.0/24; port 2 is configured for **vlan2** and subnet 172.16.0.0/24. If servers are connected to both ports, and these servers need to communicate with one another through Equalizer, you would execute the following commands to enable routing between **vlan1** and **vlan2**:

```
eqcli > vlan vlan1 permit vlan2
eqcli > vlan vlan2 permit vlan1
```

Using the **permit** command in the **vlan** context, as above, enables packet forwarding between *all* the subnets defined in the current VLAN context, and the VLAN specified as an argument to **permit**.

Routing Between Specific VLAN Subnets

In most cases, there is a one-to-one relationship between VLANs and subnets -- i.e., a VLAN in most configurations is associated with one subnet. There are, however, situations in which an administrator will associate more than one subnet with a VLAN. If multiple subnets are defined within a VLAN, you can optionally specify a subnet as an additional argument to the **permit** command, as in this example:

```
eqcli > vlan vlan1 permit vlan2:sn03
```

The above command enables ports configured for **vlan1** to route packets with a destination address on subnet **sn03** defined in **vlan2**. Packets addressed to other subnets configured on **vlan2** will be dropped.

Similarly, you'll need to specify the reverse route: let's say you only want to route packets to **vlan1** from ports configured for **vlan2** *if they originated on subnet* **sn03**. To accomplish this, you'll need to specifically add that VLAN/subnet combination to the permitted VLAN list for **vlan2**:

```
eqcli > vlan vlan2 subnet sn03 permit vlan1
```

Source IP Address for Outbound Packets

When Equalizer originates connections to other hosts (for example, when Equalizer sends out probes, queries an NTP or DNS server, etc.), the source IP address for outbound packets will be the source network that was specified in the route configured for the subnet.

Subnet Routes and Global Default Route

Each subnet has a complete routing table. There is no explicit global default route setting that applies to all subnets. To configure a global default route, you must define the same default route on all subnets.

Chapter 10
Using the GUI

Sections in this chapter include:

Logging In

The Equalizer Administrative Interface, the "GUI", is a browser based interface. In general, the GUI should function properly using any browser that is enabled for JavaScript (required). Safari® for Windows®, however, is not supported.

1. Using your browser, type one of the following into the browser's address bar:

   http://<Equalizer_IP_address>

   https://<***Load Balancer***_IP_address>

   Substitute the load balancer's IP address on a VLAN subnet that is enabled for HTTP or HTTPS, as appropriate. (see

   Equalizer displays the login screen.

2. Enter an existing login as well as the login password, and click **Login**. The **System** configuration tab on the left pane will be be expanded and the **Dashboard** screen will be displayed.

Navigating Through the Interface

The browser-based Administrative Interface (GUI), capable of operating with all commonly used web browsers, can be used to configure most of Equalizer load balancing and networking operations. If an operation can only be modified using the command line interface (CLI) it will be noted in the context of the procedures.

Some of the features and configuration screens may not be available on the following browsers:
• Internet Explorer (Verified on IE 7, 10 and 11).
• Safari for Windows

The Equalizer GUI features:

A tabbed pane on the left features groups of objects arranged in grouped hierarchical object trees. These include Equalizer’s System, Load Balance, and Log & Reports features and functions.

System

Clicking on the **System** configuration tab on the left pane provides access to **Global**,**External Services**, **Maintenance**, **Network**, and **Failover** parameters:

- Clicking on the arrow (▶) next to **Global** expands the branch and provides access to the **Dashboard**, **Alerts**, **Certificate**, **CRL**, **IP Reputation**, (Global) **Parameters**, **Smart Control**, and **SNMP** configuration displays on the right pane.
- Clicking on the arrow (▶) next to **External Services** expands the branch and provides access to **SMTP Relay** and **VLB Manager** configuration displays on the right pane.
- Clicking on the arrow (▶) next to **Maintenance** expands the branch and provides access to system **Date & Time**, **Backup & Restore**, **Licensing**, **Manage Software** (upgrade), and system **Tools** displays on the right pane.
- Clicking on the arrow (▶) next to **Network** expands the branch and provides access to **Interfaces**,(Link) **Aggregation**, **VLANs**, and **Tunnels** displays on the right pane.
  - Clicking on each **VLAN** will display the **Port Status** and **Type** tab for the selected VLAN.
  - Clicking on the arrow (▶) for each VLAN expands the branch to display the configured subnets.
  - Click on each subnet to display the subnet configuration for each.
  - Right-clicking on the VLAN label displays the **Add VLAN** command.
  - Clicking on the arrow (▶) beside **Tunnels** displays all of the defined IPv6 tunnels.
  - Clicking on each Tunnel displays the **Tunnel Configuration** display on the right pane.
  - Right-clicking on the **Tunnel** label displays the **Add Tunnel** command.
- Clicking on the arrow (▶) next to **Failover** expands the branch and displays the **Peer > Summary** screen on the right pane which displays failover status as well as a tab that displays VLAN

subnet heartbeating status. Clicking each peer in the expanded branch will display each Peer's configuration display on the right pane. Right-clicking on **Peers** displays the **Add New Failover Peer** command.

Load Balance

Clicking on the **Load Balance** configuration tab on the left pane provides access to Equalizer's load balancing objects and their parameters:

Clusters

- Clicking on the **Clusters** label displays the **Cluster Summary** screen that features an configuration tab listing of all configured clusters on the right pane. Clicking the plus sign (+) next to a cluster name displays summary information about the cluster including connection information for L4 clusters and connection and transaction information for L7 clusters.
- Clicking on the arrow (▶) expands the branch to display all configured clusters. Click on each cluster and the **Configuration > Summary** screens are displayed on the right pane.
- Right-clicking on the **Cluster** label displays the **Add Cluster** command.

Server Pools

- Clicking on the **Server Pools** label displays the **Server Pool Summary** screen that features an configuration tab listing of the configured server pools and their status. Double clicking the a server pool name opens a list of the currently defined server pools and status.
- Clicking on the arrow (▶) expands the branch to display all configured server pools. Click on each server pool and the configuration displays are available on the right pane.
- Clicking on the arrow (▶) for each Server Pool expands the branch to display all configured server instances. Click on each server instance to display the **Server Instance Configuration Summary** display on the right pane that displays active connection information as well as parameters and a graphical display of Server Pool traffic from the previous thirty minutes.
- Right-clicking on the **Server Pool** label displays the **Add Server Pool** command.

Servers

- Clicking on the arrow (▶) expands the branch to display all configured servers. Click on each server and the configuration displays are available on the right pane.
- Click on each server to display the **Server Configuration Summary** screen that displays active connection information as well as parameters and a graphical display of Server traffic from the previous thirty minutes.
- Right-clicking on the **Server** label displays the **Add Server** command.

Health Checks

- Clicking on the arrow (▶) expands the branch to display all configured health checks. Click on each health check and the configuration displays are available on the right pane.
- Click on each health check to display the **Configuration Settings** screen that displays parameters that displays the configured parameters and timers as well as a **Disable** option.
- Right-clicking on the **Health Checks**label displays the **Add Health Check** command.
- Right-clicking on each health check displays the **Delete Health Check** command.

Responders

- Clicking on the arrow (▶) expands this branch to display all of the configured Responders.
- Right-clicking on the **Responders** label displays the **Add Responder** command.

Link Load Balance

Clicking on the **Link Load Balance** configuration tab provides access to the **Outbound** and **Inbound** Link Load Balancing configuration screens on the right pane.

Outbound

- Clicking on the arrow (▶) beside **Outbound** expands the branch to display **Gateways** and **Groups**.
- Clicking on **Gateways** displays the list of configured **Link Load Balancing Gateways** on the right pane.
- Clicking on **Groups** displays the list of **Outbound Link Load Balancing Groups** on the right pane.

Inbound

- Clicking on the arrow (▶) beside **Inbound** expands the branch to display the list of **Inbound Link Load Balancing Groups**.

Global Load Balance

Clicking on the **Global Load Balance** configuration tab provides access to the Envoy geographic load balancing feature parameters:

Geoclusters

- Clicking on the arrow (▶) expands this branch to display all of the configured **GeoClusters**.
- Right-clicking on the **GeoCluster** label displays the **Add GeoCluster** command.
- Clicking on the arrow (▶) beside **GeoClusters** expands the branch to display all of the defined **GeoSite Instances** for the **GeoCluster**.
- Right-clicking on each defined **GeoCluster** displays the **Add GeoSite Instance** command.

Geosites

- Clicking on the arrow (▶) expands this branch to display all of the configured **GeoSites.**
- Clicking on each **GeoSite** displays the **GeoSite Configuration** screen on the right pane.
- Clicking on the arrow (▶) beside each **GeoSite** expands this branch to display all of the configured **GeoSite Resources.**
- Right-clicking on any **GeoSite** label displays the **Add GeoSite Resource** command.
- Clicking on each **GeoSite Resource** displays the **GeoSite Resource Configuration Required** screen on the right pane.

Log & Reports

Clicking on the **Log & Reports** configuration tab provides access to Equalizer's Event Log and Remote Sys log and CPU & Memory usage displays:

Logging

- Clicking on the arrow (▶) beside **Logging** expands the branch to display **Event Log** and **Remote Sys Log.**
- Clicking **Event Log** displays events for each element configured on Equalizer on the right pane. This includes **Clusters**, **Server Pools**, **Servers** and **Responders.**
- Clicking **Remote Syslog** displays the **Remote Syslog** screen on the right pane. It allows you to specify a remote Syslog Server and to enable the logging of events for this remote host.

Reporting

- Clicking on the arrow (▶) beside **Reporting** expands the branch to access **CPU & Memory** display.
- Clicking **CPU & Memory** displays the current and 60-minute averages of CPU Consumption percentage and Memory Utilization in Mb on the right pane

Help Buttons/Options

Log Out	Logs you out of the GUI.
Help	Displays a sub-menu of commands
Context Help	Displays the section in the Help that corresponds to the screen currently displayed in the right frame.
About	Opens the **Welcome** screen (also displayed when you first log into the GUI).

Management Tabs/Dialogue Area

The right hand side of the GUI initially displays the **Welcome** screen however it is designed to display all configuration, management and dialogue associated with the objects in the left navigational pane.

Click on any item in the left pane, or right click to choose a command for that object. The right pane will display the management details for the object or the appropriate command dialog.

The easy-to-use management tabs organize configuration information into forms and tables that make configuring Equalizer simple.

Entering Names for Load Balancing Objects

Equalizer identifies administrative objects, such as clusters and servers, by name. For example, object names and icons are displayed in a hierarchy in the GUIs left frame as described earlier in this chapter. Keep in mind the following guidelines when typing in a name for an object:

- The characters used in names are limited to standard ASCII letters ("A" through "Z" and "a" through "z"), numbers (0 through 9), and the characters "**.**" (period), "**-**" (dash) and "**_**" (underscore).
- The first character in a name must be a letter.
- Names can be at most 47 characters long.
- The readability of lists presented in the interface is increased by using short names that use as many unique characters at the beginning of the name as possible.

Using the WebHelp

Installed on your Equalizer is an HTML-based WebHelp system that is fully functional in all web browsers. It provides descriptions of how to manage EQ/OS 10 through the Command Line Interface (CLI) and the Graphical User Interface (GUI).

The PDF file of the *Equalizer Administration Guide* is still available for download from the EQ OS 10 Support Page. It is synchronized with the same revision as the WebHelp and contains the same information.

The Equalizer GUI features context-sensitive help. When you click on the **Help** button and select **Context Help**, or simply press **F1** on your keyboard, this WebHelp system will be activated in a new browser window. If you are currently browsing an Equalizer configuration screen and select **Context Help** or press **F1**, the help topic associated with the configuration screen will be displayed.

The following is an overall description of the WebHelp workspace and instructions for its use.

Table of Contents

Select the Table of Contents configuration tab to display of Table of Context. Click on the ■ icon to expand each branch.

Breadcrumb Trail

Displays a "trail of breadcrumbs" composed of the table of contents (Table of Contents) entries above the current topic in the Table of Contents hierarchy.

Search Open Topic

This text entry box is where you can enter a search term to search the open topic for specific details. Click on 🔍 after you have entered a search term.

Toolbar

The toolbar contains buttons for quick navigation, display options, topic printing, highlighting and a search area. Enter a search term in this box and the open topic will be searched. If the search yields results, they will be highlighted on the page.

- Click on after you have entered a **Topic Search** item in the search box.
- Click on if you would like to remove the search highlighting after you have used the search feature
- Click on or to navigate to the previous or next topic of a viewing sequence.
- Click on to return to the WebHelp home page.
- Click on or to navigate to the previous or next topic in the Table of Contents.
- Click on to hide the navigation panel on the left.
- Click on or to collapse or expand to topic branches in the Table of Contents.
- Click on to print the selected topic.

Help Topic Display

This is the area where the selected topic's content is displayed.

Glossary

Select the **Glossary** configuration tab to access a glossary of load balancing and Equalizer-specific terminology. Click on each term to display a definition.

Search All Topics

Click on the **Search All Topics** configuration tab to open a Search pane. Enter a term in the at the top of the pane and click on **Search**. A list of results from the entire WebHelp system will be displayed.

Click on each item in the list to navigate to the applicable topic in the WebHelp. All of the found terms will be highlighted.

Chapter 11

System Settings

Sections in this chapter include:

Global Settings

The **Global** grouping of parameters is available within the **System** configuration tab. After logging into the GUI, Click on the **System** configuration tab which should be open by default. Then click on the arrow (►) beside **Global** to expand the branch. The parameters that can be viewed/added/modified on this branch include:

1. Dashboard
2. Alerts
3. Smart Control
4. (Global) Parameters
5. SNMP
6. Certificates
7. CRL

When you select each item a different displayed will be visible in the right frame of the GUI.

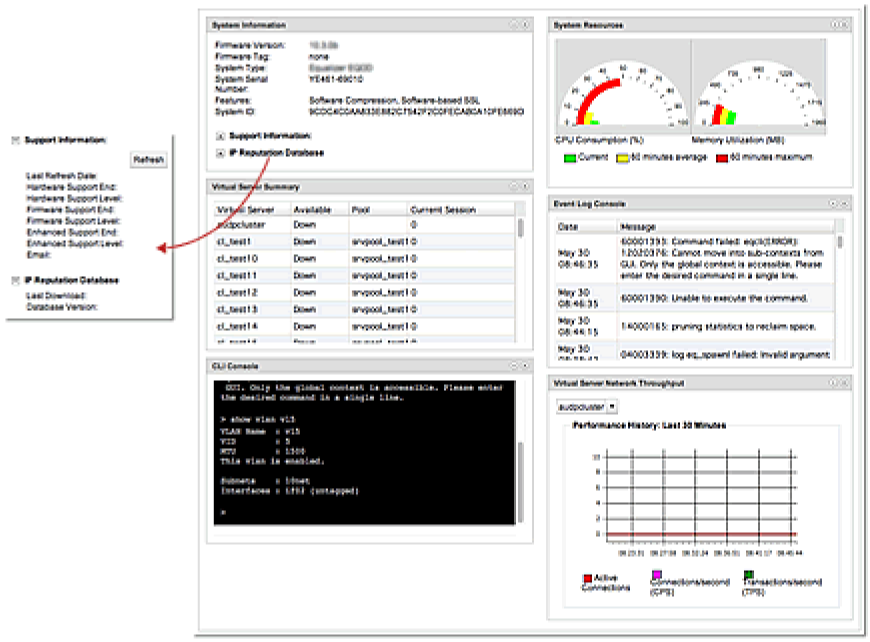
Refer to "Global Parameters" on page 157 and "Global Commands" on page 164 for Global Parameter details using the CLI.

Dashboard

The Dashboard is the initial screen displayed after logging in to the GUI. If it is not displayed you can also access it by clicking on the **System** configuration tab on the right navigational pane and then the arrow (▶) beside **Global** to expand the branch. When you click on **Dashboard** it will be display on the right frame.

The Dashboard contains expandable/collapsible tabs/widgets that can be removed from the display if desired.

By default, all of the widgets are expanded. Click on ▼ in the upper right corner of each tab to expand or collapse the tab. Click on the **X** to remove the tab from the Dashboard.



Last Refresh Date:
Hardware Support End:
Hardware Support Level:
Firmware Support End:
Firmware Support Level:
Enhanced Support End:
Enhanced Support Level:
Email:

Last Download:
Database Version:

Firmware Version:	[illegible]
Firmware Tag:	none
System Type:	[illegible]
System Serial Number:	YE461-69010
Features:	Software Compression, Software-based SSL
System ID:	[illegible]

Virtual Server	Available	Pool	Current Session
autocluster	Down		0
cl_test1	Down	srvpool_test1	0
cl_test10	Down	srvpool_test1	0
cl_test11	Down	srvpool_test1	0
cl_test12	Down	srvpool_test1	0
cl_test13	Down	srvpool_test1	0
cl_test14	Down	srvpool_test1	0

Date	Message
May 30 08:46:35	60001393: Command failed: eqcli(ERROR): 12020376: Cannot move into sub-contexts from GUI. Only the global context is accessible. Please enter the desired command in a single line.
May 30 08:46:35	60001390: Unable to execute the command.
May 30 08:44:15	14000165: pruning statistics to reclaim space.
May 30	04003339: log eq_spawn failed: invalid argument

```
 GUI. Only the global context is accessible. Please enter
the desired command in a single line.

> show vlan v13
VLAN Name   : v13
VID         : 3
MTU         : 1500
This vlan is enabled.

Subnets     : 10net
Interfaces  : 1f01 (untagged)

>
```

Virtual Server Summary	Displays a summary of the configured servers on the appliance as well as their availability and the associated server pools.
Even Log Console	Displays events for each element configured on the Equalizer. This includes Clusters, Server Pools, Servers, Peers, and Responders.

CLI Console	Most CLI functions can be performed at this console .
	
	Note: At this time, following commands are not available with this Dashboard widget:
	`show boot`
	`boot` *`boot options`*
	`hidden reset config`
	`hidden reset keep-license`
	`hidden show config`
	`hidden shell`
	`hidden shell admin`
	`hidden eqcollect url` *`url`* `[name` *`name`*`]`
	`hidden eqcollect local`
	`backup url` *`url`*
	`restore url` *`url`* `name` *`name`*
	`show sbr`
	`show files`
	`no files` *`filename`*
	`cfg_convert file` *`file name`* `[outfile` *`outfile name`*`]`
	`certificate` *`cert name`* `certfile edit`
	`certificate` *`cert name`* `keyfile edit`
	`resp` *`resp name`* `html edit`
	`files edit`
	`upgrade` *`upgrade path`*
	`files download`
	`files ftp`
	`certfile` *`url`*
	`keyfile` *`url`*
	`show log eq [[reverse] [lines` *`number of lines`*`]]`
	`show log sys [[reverse] [lines` *`number of lines`*`]]`
	`hidden show log dbg [[reverse] [lines` *`number of lines`*`]]`
	Note: You cannot enter into sub-contexts from this widget. Only global context is available. Commands must be entered in a single line.

License Information	Displays **Firmware Version** used, the **System Type** you are using, the **Serial Number** of the appliance, and specific features on the appliance such as Software-based SSL or hardware-based SSL
Virtual Server network Throughput	A visual display of the appliance throughput showing **Active Connections**, **Connections/second** and **Transactions/second**.This monitors the configured clusters. The drop down list selects the cluster to monitor.
System Resources	Displays **CPU Consumption** and **Memory Utilization** data. It displays maximums and the averages for the previous 60 minutes in addition to the real time traffic currently moving through the appliance.

Alerts

Refer to "Configuring Alerts" on page 816 for a description of alerts and setups.

Certificates

Each SSL certificate installed on Equalizer includes a certificate and its associated private key.

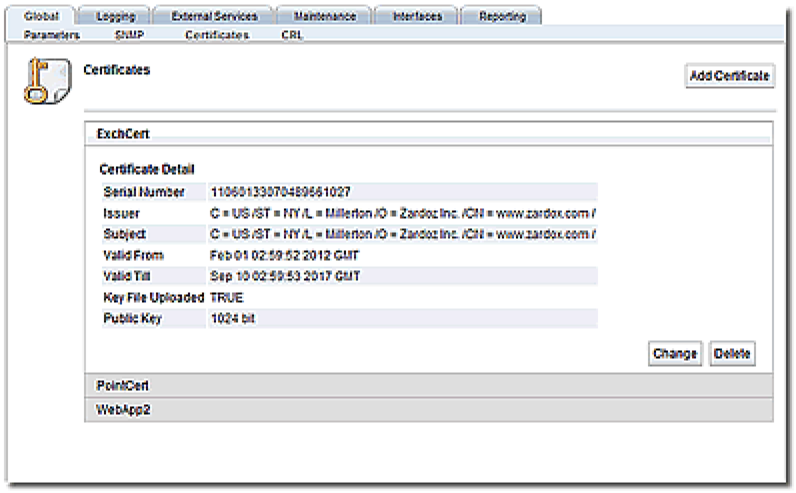
In SSL off loading, Equalizer terminates the SSL connection with the client, decrypts the client request using a certificate and key, sends the request on to the appropriate server, and encrypts the server response before forwarding it on to the client.

Certificates are uploaded to Equalizer and then associated with one or more clusters. Two types of certificates may be used to authenticate HTTPS cluster connections:

- A *cluster certificate* is required to authenticate the cluster to the client and to decrypt the client request (these are also called server certificates). For cluster certificates, both a certificate file and a private key file must be uploaded to Equalizer.
- A cluster may also be configured to ask for, or require, a client certificate -- a certificate used to authenticate the client to Equalizer. For client certificates, only a certificate file is uploaded to Equalizer (no keyfile is used).

Installing an SSL Certificate using the GUI

1. Click on the host name at the top of the left navigational pane and then click on **Global Certificates** to display the following.



Certificate Detail

Serial Number	11060133070489661027
Issuer	C = US /ST = NY /L = Millerton /O = Zardoz Inc. /CN = www.zardox.com /
Subject	C = US /ST = NY /L = Millerton /O = Zardoz Inc. /CN = www.zardox.com /
Valid From	Feb 01 02:59:52 2012 GMT
Valid Till	Sep 10 02:59:53 2017 GMT
Key File Uploaded	TRUE
Public Key	1024 bit

2. Click on **Add Certificate** to display the **Add Certificate** dialogue form .
3. Click on **Choose File** to select a locally stored **Certificate File.** Repeat the same for adding a locally stored **Key File.**
4. Click on **Commit** to save the upload the new **Certificate File** and **Key File.**

Installing an SSL Certificate using the CLI

Refer to "Certificate Commands" on page 167 for Certificate commands.

1. Create a certificate using the `certificate` command as follows:

```
eqcli > certificate certname
eqcli: 12000287: Operation successful
eqcli >
```

   where:

   certname - is the name of the certificate.

2. Add a locally stored certificate file and key file.
3. **`Commit`** to upload the new **`Certificate File`** and **`Key File`**.

Deleting a Certificate in Use

If you attempt to delete a certificate that is in use, an error will occur. You can, however use the `force` command, to remove the certificate and revert to a system certificate (`server.crt`) which is shipped with systems and cannot be deleted. This will prevent losing connectivity with the GUI.

Use the force command as follows:

```
eqcli > no certificate certname force
eqcli: 12000287: Operation successful
```

Replacing the Default Certificate, Key, and Cipherspec

Using Equalizer's Remote Management commands in the CLI, you can replace the default certificate, key, and cipher spec that are used with HTTPS services on subnets with custom certificates, keys and cipher specs.

The process includes:

- Uploading the custom certificate and key file to the file store.
- Entering the certificate (and key file) to be used with HTTPS services.
- Replacing the default cipherspec with the a custom cipher spec.
- Setting the encryption level to use in the communications between the client and the ADC.

Uploading the Custom Certificate and Key File

Enter the following to upload a certificate and key file:

1. Enter the name of the new certificate and upload it as follows:

```
eqcli > certificate certificatename certfile URL
```

   where ***URL*** downloads the **`certfile`** using **ftp://** or **http://** protocol.

2. Upload the new key file. The key file must have the same name as the certificate.

```
eqcli > certificate certificatename keyfile URL
```

   where ***URL*** downloads the **`keyfile`** using **ftp://** or **http://** protocol.

Entering the Certificate and Key file to be Used with HTTPS Services

3. Set the **`certfile`** and **`keyfile`** to use using the CLI remote management commands. The **`keyfile`** has the same name as the **`certfile`** and will be used automatically.

```
eqcli remote-mgmt certificate certificatename
```

4. Now view the remote management configuration. The example that follows shows that the `custom certificate` has been added:

```
eqcli > show remote-mgmt
Options             Value
Cipherspec          AES128-SHA:DES-CBC3-SHA:RC4-SHA:RC4-MD5:AES256-SHA:!SSLv2
Certificate         custom certificate
Protocols           tls10

eqcli >
```

Replacing the Default Cipherspec with a Custom Cipherspec

5. Enter the custom cipherspec as follows:

```
eqcli > remote-mgmt cipherspec cipherspec
```

   where ***`cipherspec`*** is the new, custom `cipherspec` to be used.

Setting the Encryption Levels

6. Configure the encryption levels that will be used in communications between the client and the ADC. The default encryption level is TLSv1.0 (`tls10`).

```
eqcli > protocol protocol
```

   where ***`protocol`*** can be **`sslv3`**, **`tls10(default)`**, **`tls11`**, or **`tls12`**. The protocols in the syntax can be delimited by "," or "|".

   You can also turn off one of the protocols in the list by prefixing with "!". For example if you have configured all of the encryption levels to be used and want to remove `tls12`, enter `eqcli >` **`protocol !tls12`**. **`tls12`** would then be removed from the list. The client and ADP will use the highest level available when multiple formats are specified.

Reapplying the Default Certificate, Cipherspec and Protocols

To reapply the defaults for Cipherspec, Certificate or Protocol, enter any of the following:

```
eqcli > no remote-mgmt {cipherspec|certificate|protocol}
```

Certificate Revocation Lists

The Certificate Revocation List (CRL) can be used to verify that the certificates used byare valid and have not been compromised. A CRL is uploaded to and then associated with one or more clusters in the cluster specific context. Whenever a certificate is used to authenticate a connection to the cluster, the CRL is checked to make sure the certificate being used has not been revoked.

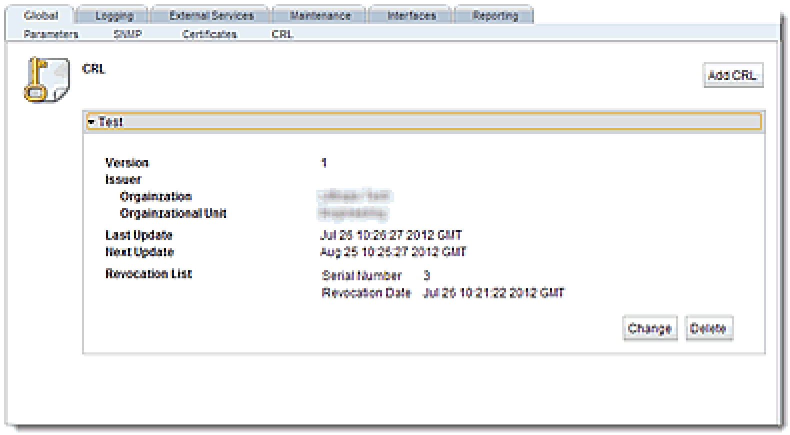
Equalizer provides support for Certificate Revocation Lists (CRLs) using a central CRL store to which CRLs can be uploaded and then associated with as many clusters as required.

If a CRL attached to a cluster was generated by a Certificate Authority (CA) different from the CA used to generate a client certificate presented when connecting to the cluster, an error will occur, The CRL and client certificate must be signed by the same CA.

Installed CRLs will be displayed in an accordion style list. Click on each list item to expand it and display the contents of the CRL.

Installing a CRL using the GUI

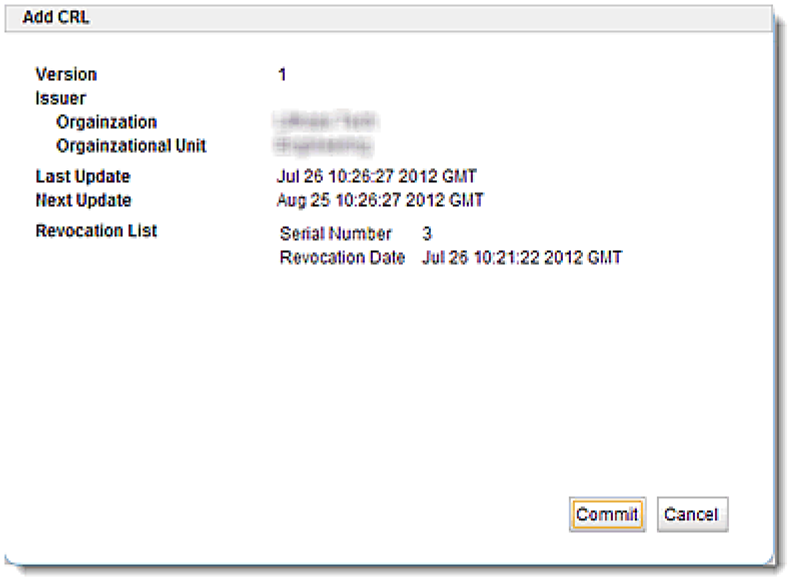
1. Click on the host name at the top of the left navigational pane and then click on **Global CRL** to display the following.



Version	1
Issuer	
Orgainzation	[redacted]
Orgainzational Unit	[redacted]
Last Update	Jul 26 10:26:27 2012 GMT
Next Update	Aug 25 10:26:27 2012 GMT
Revocation List	Serial Number 3
	Revocation Date Jul 26 10:21:22 2012 GMT

2. Click on **Add CRL** to display the **Add CRL** dialogue screen.

3. Click on **Choose File** to select a CRL file to upload to Equalizer. Select a ***.crl** file to upload, enter a **Name**, and then click on **Commit**. A confirmation screen will appear as follows:



Version	1
Issuer	
Orgainzation	
Orgainzational Unit	
Last Update	Jul 26 10:26:27 2012 GMT
Next Update	Aug 25 10:26:27 2012 GMT
Revocation List	Serial Number 3 Revocation Date Jul 26 10:21:22 2012 GMT

Click on **Commit** if the **CRL** is the one you would like to upload to Equalizer. The CRL file will be uploaded to Equalizer and will appear on the **Global > CRL** screen as shown above.

Installing a CRL using the CLI

Refer to "Certificate Revocation List Commands" on page 170 for details on using the CLI.

IP Reputation

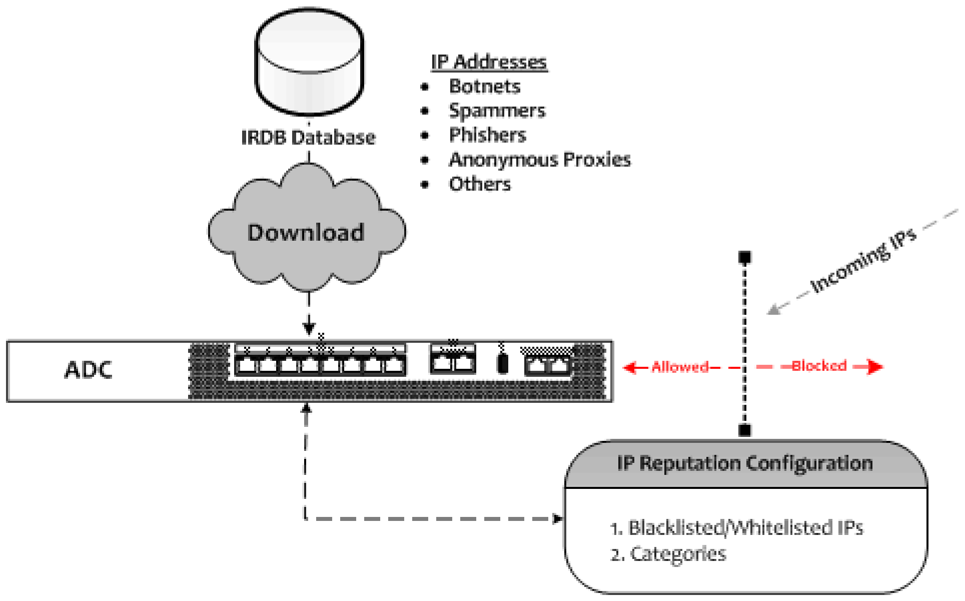
Note - IP Reputation is not supported on Equalizer E250GX, E350GX, E450GX, or E650GX.

Security threats arise from a variety of sources on the internet: botnets, spammers, phishers, etc., are all common threats that you want to keep off your network. Manually identifying suspect client IP addresses from which these threats originate is a complex task that many organizations do not have the resources to tackle. The IP Reputation feature provides you with a vigorously maintained database of IP addresses of compromised and malicious clients – as well as the infrastructure necessary to refuse those clients access to your network. It uses accurate, early, and frequently updated identification so you can block these attackers before they target your servers.

Data about dangerous clients is derived from many sources around the globe. This data is compiled into Fortinet’s IP Reputation Database (IRDB), which consists of the IP addresses of suspect clients. Clients are identified and tagged with poor reputations and included in the IRDB if they have been participating in attacks, willingly or otherwise. Configuring IP Reputation includes the following tasks:

- Enabling IP Reputation Blocking.
- Downloading the IRDB from Fortinet.
- Optionally selecting specific categories of IP addresses to block.
- Optionally adding IP addresses to a whitelist and blacklist.

The above tasks are shown graphically in the figure below:



Statistics are also generated that show all blocked IP addresses and the number of packets blocked for each IP address.

Enabling and Disabling

Note - IP Reputation commands are not available if using Equalizer E250GX, E350GX, E450GX, or E650GX.

The IP Reputation Enable/Disable flag allows you to:

1. Fetch the IRDB from Forticare and use it.
2. Configure blacklists and whitelists.

By default, IP Reputation functionality is enabled. To disable, enter:

```
eqcli > reputation disable
```

If you have previously disabled IP Reputation and need to re-enable it, enter:

```
eqcli > reputation enable
```

To enable or disable IP Reputation using the GUI proceed with the following:

Note - The **IP Reputation** object on the **Global** branch of the left navigational pane will not be visible if using Equalizer E250GX, E350GX, E450GX, or E650GX. In addition, the **System Information** pane on the GUI Dashboard, will not display IP Reputation Database information.

1. Click on the **System** configuration tab on the left navigational pane.
2. Expand the **Global** tree. Select **IP Reputation** to display the **IP Reputation** screen .
3. Using the **IP Reputation Tracking** radio buttons to turn IP Reputation **On** or **Off**. When **Off** is selected, no IP Reputation tracking is performed, regardless of individual settings.
4. Click on **Commit** to save the *setting.*

Downloading the IRDB Database

The IP Reputation functionality is dependent upon the IP Reputation Database (IRDB), created and managed by Fortinet. The IRDB contains IP addresses and network ID ranges (grouped into the categories described above) that pose a threat to your network. After you register your appliance with Fortinet support, you can download the database from the support site (assuming that your support contract includes IRDB access). Your appliance must have access to the internet to download the IRDB.

The IRDB is updated frequently by Fortinet and should be refreshed on a regular basis. The UI displays the date the IRDB was last updated. Statistics from blocked or passed IPs used with previous IRDB downloads will not be removed when an updated IRDB is downloaded.

The IRDB database can be downloaded using two methods:

- using controls in the UI to download the database on demand
- using an automated Smart Control (See "Smart Control Overview" on page 780) to download the database regularly

In order to download the IRDB database, verify that IRIS Service (IP Reputation Intelligence Service) has been enabled for your registered product on the Fortinet Support site. This will appear in the Product Entitlements section of the product.page.

Manual Download

Note - Your appliance must have internet access to download the IRDB database.

You will need to download the IRDB database before IP Reputation is fully functional.To verify that the IRDB database has been downloaded, the current IRDB date, and the version currently installed, enter the following using the CLI:

```
eqcli > show reputation
Current IRDB Version : 05000000IRFW00303-00001.00690-0000000000
Current IRDB Date    : 05/10/2014 10:41:22
IP Reputation        : Enabled
Category          Action
botnet            block
anonymous_proxy   block
phishing          block
spam              block
others            block
```

If the IRDB has not been downloaded, such as if you were configuring a new appliance, the Current IRDB Version will appear without a version. Also, if you attempt any of the configuration commands described in the following section, an error message will appear.

There are two methods of uploading an updated IRDB to your appliance. The first method is a direct download of a current IRDB from Fortinet and "fetching" it using the CLI command line. This method requires direct, real-time connectivity between your appliance and Fortinet support servers. The second method does not require your system to have direct connectivity with the Fortinet support servers. However, you will need to have internet access to the Fortinet Support page so that you can download a file, and place the downloaded file containing the IRDB on a host from which your appliance can connect via FTP.

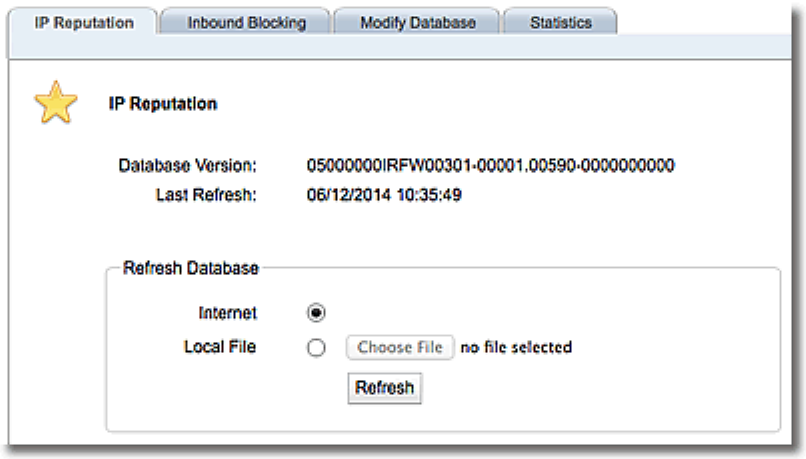
Direct download of the IRDB database using the CLI:

Enter the following in the CLI:

```
eqcli > reputation fetch
eqcli: 12000287: Operation successful
```

Direct download of the IRDB database using the GUI:

1. On the left navigational pane, select **System > Global > IP Reputation** to display the IP Reputation screen below.



Database Version: 05000000IRFW00301-00001.00590-0000000000
Last Refresh: 06/12/2014 10:35:49

Refresh Database
Internet [✓]
Local File

2. Select the **Internet** radio button.
3. Click on the **Refresh** button to download the latest database.

Downloading a .pkg file and uploading the IRDB database

Using this method requires you to download a `.pkg` file from Fortinet Support. The `.pkg` file you download contains the current IRDB. You will need to upload the downloaded file to an FTP site from which Equalizer can access it.

1. Access the Fortinet Support site and log in. You must have previously registered your product to access IP Reputation updates.
2. Scroll to the **Download** area of the page and click on **Service Updates.**
3. Click on **IP Reputation Updates** on the left navigational pane. A list of update files will appear on the right pane.
4. Click on the **FWB** file and follow the prompts to download the file. The downloaded file name will have a prefix of "`IRISUpdate`" with a `.pkg` extension.

Using the CLI:

1. Upload the `.pkg` file to a host system. The file must be placed in a directory from which Equalizer can connect via FTP.
2. Enter the following to upload the `.pkg` file to your Equalizer:

```
eqcli> reputation load  FTP Address .pkg File name
```

   The CLI command will:

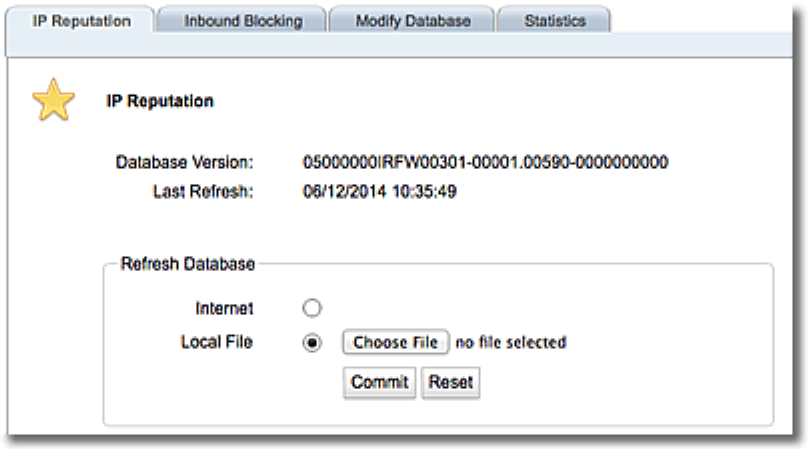
   - Transfer this file via FTP from the host system to Equalizer.
   - Translate the downloaded file to an IRDB-format (`*.db` file).
   - Install the IRDB as the current IRDB.

3. Enter `eqcli > show reputation` to verify the IRDB Version and Date.

```
eqcli > show reputation
Current IRDB Version : 05000000IRFW00301-00001.00590-0000000000
Current IRDB Date    : 06/12/2014 10:35:49
IP Reputation        : Enabled
Category         Action
botnet           block
anonymous_proxy  block
phishing         block
spam             block
others           block
```

Using the GUI:

1. Download the `.pkg` file as described above.
2. On the left navigational pane, select **System > Global > IP Reputation** to display the following.



IP Reputation

Database Version: 05000000IRFW00301-00001.00590-0000000000
Last Refresh: 06/12/2014 10:35:49

Refresh Database

Internet ○
Local File ● Choose File no file selected

3. Select the **Local File** radio button and click on **Choose File**.
4. Follow the prompts to upload the .pkg file.
5. Click on **Commit** to save the new database.

Using a Smart Control For Regularly Scheduled IRDB Downloads

You can configure a Smart Control to automatically download the IRDB at a regularly scheduled time.

To configure a Smart Control to download the IRDB database using the CLI:

1. Enter the following to assign a name to the Smart Control.

```
eqcli > smart_control fetch_irdb
```

   The CLI will enter the smart control context.

2. Activate the editor by entering the following in the smart control context.**edit** invokes the script editor to enter/create the desired script.

```
eqcli sc-fet*> script edit
```

3. Construct a script in the editor to fetch the IRDB database.

```
^[ (escape) menu  ^e search prompt  ^y delete line    ^u up     ^p prev page
^a ascii code     ^x search         ^z undelete line  ^d down   ^n next page
^b bottom of text ^g begin of line  ^w delete word    ^l left
^t top of text    ^o end of line    ^v undelete word  ^r right
^c command        ^k delete char    ^f undelete char      ESC-Enter: exit ee
L: 1 C: 30 ===================================================================
adc::cli("reputation fetch");
```

4. Exit the editor and be sure to save the script to the datastore.

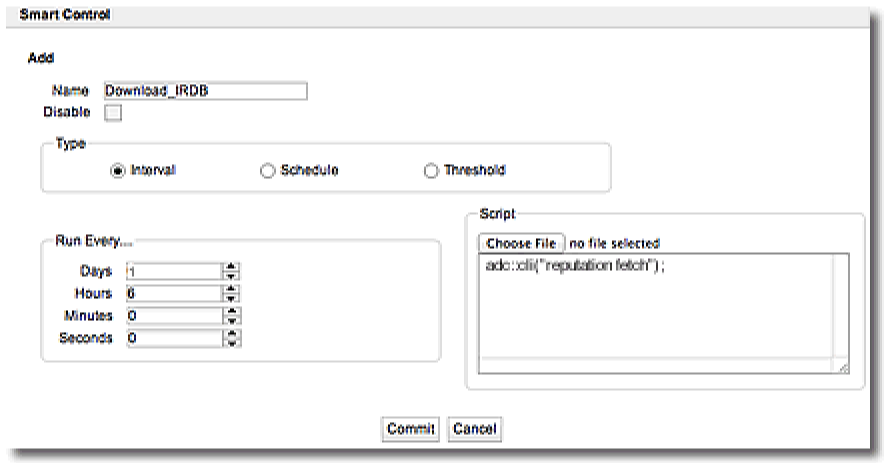
5. The Smart Control should be run at a regular interval. This is entered in seconds. In the example below, it is configured to run every 6 hours (21600 seconds). A 6 hour interval is recommended, however, you can create an interval that best fits your needs. Enter the following.

```
eqcli sc-fet*> interval"21600"
```

   You can also enter other details in the ***schedule string***. Refer to the CLI context help or "Smart Control" on page 217 for additional information.

To configure a Smart Control to download the IRDB database using the GUI:

1. Click on the **System Configuration** tab on the left navigational pane and expand the **Global** branch.
2. Select **Smart Control** to display the Smart Control display on the right.
3. Click on + to create a new Smart Control. The following will be displayed.



Smart Control

Add

Name: Download_IRDB

Disable: [ ]

Type: [✓ Interval] Schedule Threshold

Run Every...

Days	1
Hours	6
Minutes	0
Seconds	0

Script: Choose File no file selected

adc::cli("reputation fetch");

4. On the **Smart Control** configuration screen:
   a. Enter a **Name** for the Smart Control.
   b. Enter the **Type** of Smart Control. For example, in the configuration above, the Smart Control will be run at 6-hour intervals. Select the **Interval** option and then select a time interval in the **Run Every** area.
   c. Enter a **Script** that will fetch the IRDB database. In the example above `adc::cli ("reputation fetch")` is used. Refer to "Smart Control" on page 217 for additional information on entering scripts or uploading local scripts.
5. Click on **Commit** to save the IRDB Download Smart Control.

Whitelisting & Blacklisting Equalizers Configured in Failover

Whitelists and blacklists are synched across all the peers in a failover configuration.

Blacklisting Categories

The following are the categories of potential malicious attackers:

- **Botnet** - "botnet"is a merged word, derived from "robot" and "network". Sometimes called a "zombie army", it represents a number of computers on the web that, although their owners are unaware of it, have been set up to forward transmissions such as spam or viruses to other computers.
- **Phishing** - this is the act of trying to obtain confidential information from web users, typically by sending an e-mail that looks as if it is from a legitimate organization, but contains a link to a fake website that replicates the real one.
- **Anonymous Proxy** - anonymous proxies provide anonymity - users accessing websites through an anonymous proxy can't easily be traced back to their original IP.They are typically used to circumvent security policies, allowing users to access prohibited recreational, adult or other non-business sites by tunneling this traffic over a regular or encrypted HTTP session.
- **Spam** - spam is normally defined as unsolicited, electronic "junk mail".
- **Other** - this is a category of IP addresses that has been identified as potentially malicious, however, has not been categorized within the IRDB as one of the above categories.

By default, categories are not blocked and are allowed to pass. It is possible that you may want to block an entire category or IPs in the IRDB.

Enter the category using the following format. Options are **`botnet`**, **`anonymous_proxy`**, **`phishing`**, **`spam`**, or **`others`**:

```
eqcli > reputation block category
```

If, for example, you blacklist the **botnet** category, the following will be displayed when you verify:

```
eqcli > show reputation category
Name             Blocked Direction
Botnet           inbound
Anonymous_Proxy  none
Phishing         none
Spam             none
Others           none
eqcli >
```

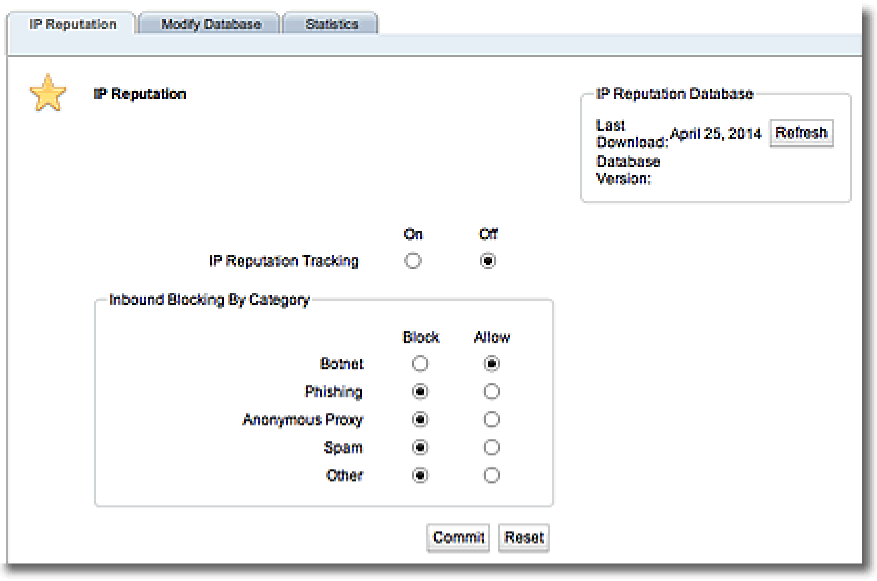
Removing Categories from the Blacklist

The following format is used to remove previously configured categories from blacklists.

```
eqcli > reputation pass category
```

To blacklist a category using the GUI:

1. Select **System > Global > IP Reputation** .on the left navigational pane to display the **IP Reputation** screen.



Last Download: April 25, 2014
Database Version:

	On	Off
IP Reputation Tracking	○	●

Inbound Blocking By Category

	Block	Allow
Botnet	○	●
Phishing	●	○
Anonymous Proxy	●	○
Spam	●	○
Other	●	○

Note - If **IP Reputation Tracking** has been disabled, the **Inbound Blocking by Category** are will be "grayed out" and non-functional.

2. Use the radio buttons to **Block** or **Allow** inbound **Botnet**, **Phishing**, **Anonymous Proxy**, **Spam** or **Other** that have been identified as malicious in the IRDB database.
3. Click on **Commit** to save your settings.

Modifying the Database

Besides enabling and disabling IP Reputation processing as a whole (See above), you can also enable and disable IP reputation for each for specific IP addresses. This is typically called “blacklisting” and “whitelisting”:

- Blacklisting: specifying a list of IP addresses not contained in the IRDB that will be blocked.
- Whitelisting: specifying a list of IP addresses contained in the IRDB that will never be blocked.

Blacklisting Client IP Addresses

It is possible that you may want to block one or more IP addresses that do not appear in the IRDB. You can essentially add IP addresses to the IRDB by creating a "blacklist", or list of IP addresses that will be blocked as if they appeared in the IRDB. The **block** command blocks all IRDB inbound IPs in the specified category or list of IP addresses.

The following examples demonstrated how to block a single IP or a list of IPs. A list is comma separated as shown in the example below:

```
eqcli > reputation blacklist 172.16.1.170,172.16.1.175,172.16.3.245
```

Verify your entry by entering:

```
eqcli > show reputation blacklist
Blocked IP Name  Start IP Address  End IP Address    Blocked Direction
172.16.1.170     172.16.1.170      172.16.1.170      inbound
172.16.1.175     172.16.1.175      172.16.1.175      inbound
172.16.3.245     172.16.3.245      172.16.3.245      inbound

eqcli >
```

You could also enter a range of IP addresses to block. If, for example, you enter **10.0.0.5 - 10.0.0.11**, all the addresses from 10.0.0.5 to 10.0.0.11 will be blocked.The format below is used:

```
eqcli > reputation blacklist start IP-end IP
```

You can also enter a range of ip addresses using CIDR notation. For example, you could enter the following:

```
eqcli > reputation blacklist 192.168.100.0/22
```

This would encompass the 1024 addresses from 192.168.100.0 to 192.168.103.255. To verify the addresses that are blocked enter:

```
eqcli > show reputation blacklist

Blocked IP Name  Start IP Address  End IP Address   Blocked Direction
192.168.100.0    192.168.100.0     192.168.103.255  inbound

eqcli >
```

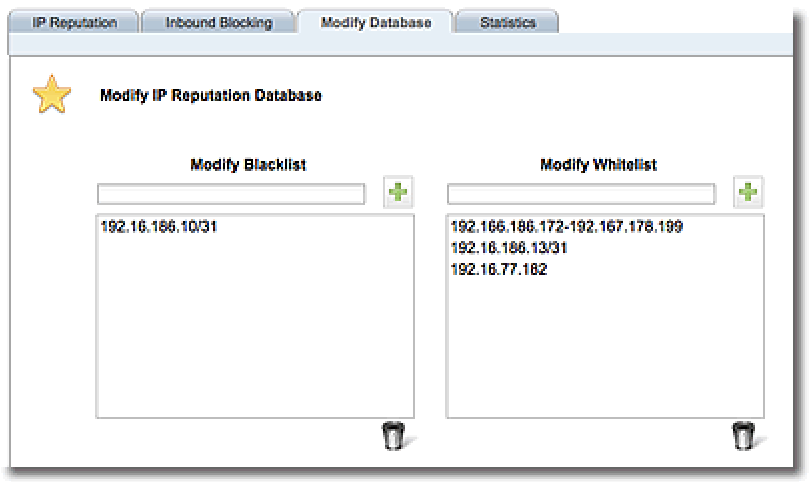
Removing IP Addresses from the Blacklist

The following format is used to remove previously configured IP addresses or categories from blacklists.

```
eqcli > no reputation {CIDR List|IP address}
```

To blacklist IP addresses using the GUI:

1. Select **System > Global > IP Reputation**on the left navigational pane.
2. Click on the **Modify Database** tab on the right. The following will be displayed:



192.16.186.10/31

192.166.186.172-192.167.178.199
192.16.186.13/31
192.16.77.182

3. Enter IP addresses in the Modify Blacklist text box and click on + to add them to the list below.
   - You could enter individual IP addresses.
   - You could also enter a range of IP addresses to block. If, for example, you enter 10.0.0.5 - 10.0.0.11, all the addresses from 10.0.0.5 to 10.0.0.11 will be blocked.

- You can also enter a range of ip addresses using CIDR notation. If, for example, you enter 192.168.100.0/22, this would encompass the 1024 addresses from 192.168.100.0 to 192.168.103.255.

4. Remove any of the addresses from the blacklist by selecting the IP addresses, or range of IP addresses in the list and click on the trash icon.

Whitelisting Client IP Addresses

As described above, a whitelist is a list of IP addresses or categories that will be allowed to pass, regardless of whether they are identified as potentially malicious in the IRDB database. The command format used to "block" addresses is similar to the format used to "pass" addresses (explained above). The **pass** command permits inbound IP addresses found in the IRDB database to access clusters defined on the ADC.

You can whitelist a single IP or a list of IPs. The list is comma separated. In the example below a list of IP addresses is shown:

```
eqcli > reputation whitelist 172.16.1.170,172.16.1.175,172.16.3.245
```

Verify your entry by entering:

```
eqcli > show reputation whitelist
Allowed IP Name  Start IP Address  End IP Address    Allowed Direction
172.16.1.170     172.16.1.170      172.16.1.170      inbound
172.16.1.175     172.16.1.175      172.16.1.175      inbound
172.16.3.245     172.16.3.245      172.16.3.245      inbound

eqcli >
```

You could also enter a range of IP addresses to pass .If, for example, you enter **10.0.0.5 - 10.0.0.11**, all of the addresses from 10.0.0.5 to 10.0.0.11 will be passed. The following format is used:

```
eqcli > reputation pass start IP-end IP
```

You can enter a range of ip addresses using CIDR notation. For example, you could enter the following:

```
eqcli > reputation pass 192.168.100.0/22
```

This would encompass the 1024 addresses from 192.168.100.0 to 192.168.103.255. To verify the addresses that are whitelisted enter:

```
eqcli > show reputation whitelist
Allowed IP Name  Start IP Address  End IP Address    Allowed Direction
192.168.100.0    192.168.100.0     192.168.103.255   inbound

eqcli >
```

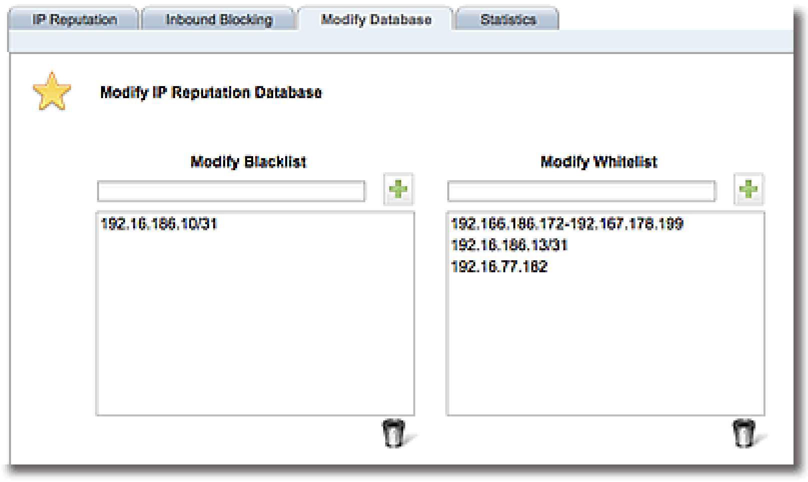
Removing IP Addresses from the Whitelist using the CLI

The following format is used to remove previously configured IP addresses from whitelists.

```
eqcli > no reputation pass {CIDR List|IP address}
```

Whitelisting IP Addresses using the GUI

1. Select **System > Global > IP Reputation** on the left navigational pane.
2. Click on the **Modify Database** tab on the right. The following will be displayed:



192.16.186.10/31

192.166.186.172-192.167.178.199
192.16.186.13/31
192.16.77.182

3. Enter IP addresses in the Modify Whitelist text box and click on + to add them to the list below.

   a. You could enter individual IP addresses that will be allowed.

   b. You could also enter a range of IP addresses to allow. If, for example, you enter `10.0.0.5 - 10.0.0.11`, all the addresses from 10.0.0.5 to 10.0.0.11 will be allowed.

   c. You can also enter a range of ip addresses using CIDR notation. If, for example, you enter **192.168.100.0/22**, this would encompass the 1024 addresses from 192.168.100.0 to 192.168.103.255.

4. Remove any of the addresses from the whitelist by selecting the IP addresses, or range of IP addresses in the list and click on the trash icon.

Displaying the Database IPs for Each Category

You can display the IP addresses that have been identified as potentially malicious in the IRDB for each category.

Using the CLI

To view the IP addresses in categories using the CLI, enter:

```
eqcli > show reputation category category
```

The example below shows 16 IRDB IP addresses for the `botnet` category.

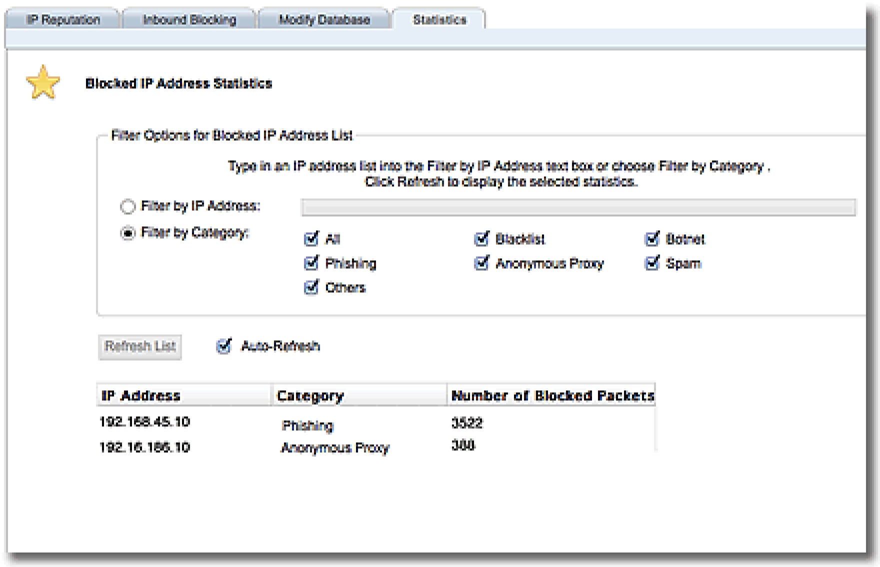
```
eqcli > show reputation category botnet
Category 'botnet' has 16 IP addresses:
1.0.243.254
1.11.85.56
1.11.149.254
1.11.173.96
1.11.183.116
1.11.195.119
1.22.8.216
1.22.16.180
1.22.17.79
1.22.19.49
1.22.28.30
1.22.35.88
1.22.38.1
1.22.38.241
1.22.71.16
1.22.71.230
```

Using the GUI

To view the IP addresses in categories using the GUI, enter the **Global > IP Reputation > Modify Database** screen and click on the **Display IP Reputation Database** button. A display similar to the one below will be displayed. IP addresses that have been identified as potentially malicious in the IRDB for each category as well as blocked statistics are displayed.

Click on the appropriate category option to select a category's IP addresses.

You can also display blocked statistics for a specific IP address by selecting the Filter by IP Address option and entering an IP address in the space provided. You can enter a range of IP addresses or CIDR format as described previously.



Blocked IP Address Statistics

Filter Options for Blocked IP Address List

Type in an IP address list into the Filter by IP Address text box or choose Filter by Category .
Click Refresh to display the selected statistics.

Filter by IP Address: [✓ Filter by Category:]

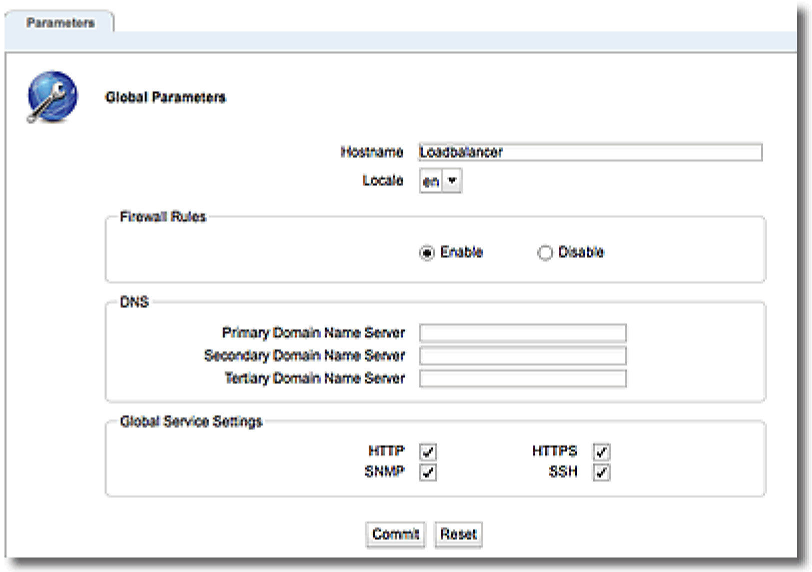
✓ All ✓ Blacklist ✓ Botnet
✓ Phishing ✓ Anonymous Proxy ✓ Spam
✓ Others

✓ Auto-Refresh

IP Address	Category	Number of Blocked Packets
192.168.45.10	Phishing	3522
192.16.186.10	Anonymous Proxy	388

Parameters

1. Click on the **System** configuration tab.
2. Click on the arrow (►) beside Global to expand the branch.
3. Click on **Parameters** to display the **Global Parameters** screen on the right frame.



Hostname: Loadbalancer
Locale: en
Firewall Rules: [✓ Enable] Disable
DNS
Primary Domain Name Server
Secondary Domain Name Server
Tertiary Domain Name Server
Global Service Settings
HTTP ✓ HTTPS ✓
SNMP ✓ SSH ✓

The following Global Parameters are configured on this screen (tab). Click on **Commit** to save your parameters or Reset to return the default values.

Hostname	This is Equalizer's host name (default:).
Locale	Sets the Equalizer locale. **en** to set English locale. ja to set Japanese locale.

Firewall Rules

Enable/Disable	This is Enabled by default. If Disable is selected Any Permit and Deny selections on subnets in your configuration will be ignored. Refer to "IP Filter Rules" on page 136 for additional information.

DNS Section

Domain Name Server (Primary, Secondary, or Tertiary)	If using a Domain Name Server, the Domain Name Server Equalizer will use.

ICMP Health Checks Section

ICMP Relaxed Probe (flag)	Global Enable/Disable ICMP relaxed probing. When enabled, if a server probes "down", but has not previously probed "up", it will be marked "up". Setting of this flag prevents the sudden reporting of servers as being "down" following an upgrade.
ICMP Probe Maximum Tries	Enables probing servers using ICMP echo (ping) probes. These probes are 5 seconds apart. If a server does not respond to an ICMP probe and it has attempted to probe the number of times specified, it is marked *down* only if there are no other probes (TCP, ACV, or server agent) active for the cluster. This value must be greater than 1.
ICMP Probe Interval	A timer specifying the length of time (in seconds) during which a successful server probe must occur, or the server is marked "down". At least one ICMP probe of a server must have succeeded since the last Equalizer reboot, or failed ICMP probes for the server will be ignored and the server will be marked "UP.

Global Service Settings Flags

The permitted VLAN services are **HTTP**, **HTTPS**, **SNMP**, **SSH** are set globally, here. Use the check boxes to enable.

Server Side Encryption

Server Side Encryption (SSE) provides the ability to configure a cluster, server, or match rule so that traffic between the Equalizer and servers is encrypted using SSL/TLS.

Refer to "Server Side Encryption" on page 396 for a description of configuring SSE using the GUI and the CLI.

Smart Control

The Smart Control feature allows you to define a common administrative function or, Smart Event that executes the function based on pre-set threshold values for system parameters and statistics. It is a method for administrators to configure the system to automatically perform functions that may be dependent on threshold values or timing.

Refer to "Smart Control Overview" on page 780 for complete descriptions of this function.

SNMP

The Simple Network Management Protocol (SNMP) is an internet standard that allows a management station to monitor the status of a device over the network. SNMP organizes information about the Equalizer and provides a standard way to help gather that information. Using SNMP requires:

- An SNMP agent running on the system to be monitored.
- A Management information Base (MIB) database on the system to be monitored.
- An SNMP management station running on the same or another system.

An SNMP agent and MIB databases are provided for SNMPv1 and SNMPv2c. If using a MIB browser, use SNMPv2c to ensure returned statistics.

A management station is not provided with Equalizer and must be obtained from a third party supplier. The management station is often used primarily to browse through the MIB tree, and so is sometimes called a MIB browser. One such management station that is available in a free personal edition is the iReasoning MIB Browser, available from www.ireasoning.com.

A MIB database is a hierarchical tree of variables whose values describe the state of the monitored device. A management station that want to browse the MIB database on a device sends a request to the SNMP agent running on the device. The agent queries the MIB database for the variables requested by the management station, and then sends a reply to the management station.

With SNMP, you can monitor the following information from the Equalizer MIBs:

Static configuration information, such as:

- Device name and Model
- Software version
- Internal and External IP addresses and netmasks
- Default gateway
- Failover alias

Equalizer failover details

- Sibling Name
- Sibling Status (Primary or Secondary)

Dynamic configuration information, such as:

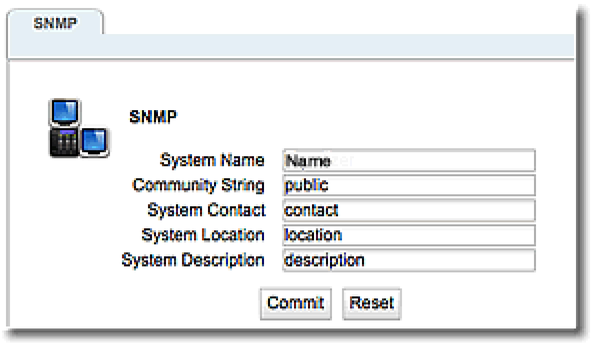
- Failover Status (Primary or Secondary)
- NAT enabled
- L4 configuration state
- L7 configuration state
- Server Health check status
- Email status notification
- Cluster parameters (timeouts, buffers)
- Server parameters Equalizer status
- L4 Statistics
- L7 Statistics Equalizer cluster
- L4 or L7 protocol of cluster

- Load balancing policy for cluster
- IP address and port (or range)
- Sticky time and inter cluster sticky
- Cookie On or Off

By default, SNMP is a globally enabled service -- meaning that it will run on any subnet that is configured to offer the SNMP service. You must specifically enable SNMP on the subnet or subnets on which you want it to listen for SNMP MIB browser and management station connections.

SNMP Parameters using the GUI

SNMP parameters are displayed on the GUI by clicking on the **System** configuration tab on the left navigational pane and then selecting **Global** to expand the branch. Click on **SNMP** to display the following:



SNMP

System Name	Name
Community String	public
System Contact	contact
System Location	location
System Description	description

The parameters are as follows:

System Name - this is the name assigned to the system.

Community String - Any SNMP management console needs to send the correct community string along with all SNMP requests. If the sent community string is not correct, Equalizer discards the request and will not respond.

System Contact - Contact is the name of the person responsible for this unit.

System Location - Location describes Equalizer’s physical location.

System Descriptions - this is the user-assigned description of Equalizer.

Click on **Commit** to save your changes.

SNMP Parameters using the CLI

Refer to "SNMP Commands" on page 219 for details.

MIB Compliance

EQ/OS 10 fully supports these proprietary Equalizer MIBs:

- CPS-EQUALIZER-v10-MIB
- CPS-REGISTRATIONS-v10-MIB

Note - In Equalizer EQ/OS 8.6 the names of the MIBs were CPS-REGISTRATIONS-MIB and CPS-Equalizer-MIB; their filenames were cpsreg.my and cpsequal.my, respectively. Substantial changes were made to the MIBs for EQ/OS 10. Therefore, the MIB names and filenames are changed for EQ/OS 10. EQ/OS 8.6 MIBS will not work with EQ/OS 10 MIBS.

EQ/OS 10 provides partial support for these standard MIBs:

RFC1213-MIB (RFC1213)

System	tcp: *tcpInSegs, tcpOutSegs, tcpRetransSegs, tcpInErrs, tcpOutRsts; and tcpConnTable*
Interfaces	udp
IP: *ipForwarding, ipDefaultTTL, ipInReceives, ipInHdrErrors, ipInAddrErrors, ipForwDatagrams, ipInUnknownProtos, ipInDiscards, ipInDelivers, ipOutRequests, ipOutDiscards, ipOutNoRoutes, ipReasmReqds, ipReasmOKs, ipReasmFails, ipFragOKs, ipFragFails, ipFragCreates; ipAddrTable and ipNetToMediaTable*	snmp
icmp	**The remaining objects are not supported**

HOST-RESOURCES-MIB (RFC2790)

hrSystemUptime	hrProcessorLoad
hrSystemDate	hrNetworkTable
hrStorage	**The remaining objects are not supported.**
hrDeviceTable	

MIB Files

MIBs are included with your appliance and updated when the firmware is updated. It is recommended that you update your SNMP Management station (MIB Browser) after updating the firmware to insure that you are browsing the current MIB tree.

All MIBs referenced by the supported MIBs are included on Equalizer.

The MIB filenames comprise the MIB name plus the filename extension ".my":

```
CPS-EQUALIZER-v10-MIB.my
CPS-REGISTRATIONS-v10-MIB.my
HOST-RESOURCES-MIB.my
HOST-RESOURCES-TYPES.my
IANAifType-MIB.my
IF-MIB.my
INET-ADDRESS-MIB.my
IP-MIB.my
RFC1155-SMI.my
RFC1213-MIB.my
SNMPv2-CONF.my
SNMPv2-MIB.my
SNMPv2-SMI.my
SNMPv2-TC.my
TCP-MIB.my
UDP-MIB.my
```

The MIB files can be downloaded from Equalizer using a browser pointed at:

```
    http://<Equalizer>/eqmanual/<mibname>.my
```

External Services

SMTP Relay

SMTP Relays are commonly used when you want to configure email alerts.

With email alerts, you be adding email addresses to the alert. Refer to "Configuring an SMTP Relay" on page 834 for additional information.

VLB Manager

In order to obtain VMware virtual machine information, Equalizer needs access information for the vCenter console (or ESX server) managing the virtual machines. To enable communication between Equalizer and a vCenter console, the **External Services VLB Manager** configuration.

Parameters

VLB Managers are configured using the parameters described in the VLB Manager Parameters table below.

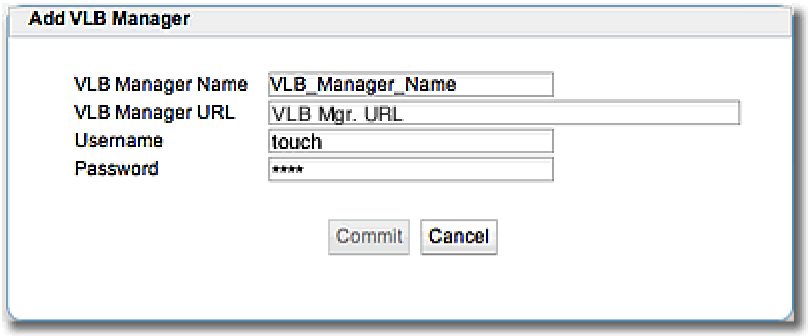
VLB Manager Parameters

GUI Parameters (CLI Parameter)	Description
VLB Manager Name (`name`)	A name for the VLB Manager. This can be up to 51 ASCII characters that includes: letters, numbers, period (.), dash (-), asterisk (*), collon (:), "at" sign (@), or underscore (_).
VLB Manager URL (`URL`)	The URL configured on the system running vCenter (or on an ESX Server) for VMware API connections. By default, this is an https:// URL using the IP address of the vCenter system followed by /sdk, as in: **`https://192.168.1.50/sdk`** The VLB Manager URL parameter cannot contain (or resolve to) an IPv6 address. Only an IPv4 address can be used. I
Username (`username`)	The VMware user account that you normally use to log into the vCenter or ESX Server that manages your VMware configuration.
Password (`password`)	The password for your VMware user account. (Note that this text box is blank when you open the tab, even if a password has been previously saved.)
Flags	
Enable/Disable (`enable/disable`)	When enabled, this allows Equalizer to communicate with virtual machines and to log into VMware automatically and enable virtual machines in clusters. If the **Disable** option is selected, no connections to Virtual Center will be made.

VLB Manager using the GUI

Create a VLB Manager as an External Service as follows:

1. Log in to the GUI.
2. Select **System** > **External Services** > **VLB Manager**.
3. Click on the "+" sign on the right configuration pane to add a new VLB Manager. The **Add VLB Manager** screen is used to set up communication login credentials to VMware using saved configurations that allow easy communication between Equalizer and VMware.
4. Log in to the GUI.
5. Select **System** > **External Services** > **VLB Manager**.
6. Click on the "+" sign on the right configuration pane to add a new VLB Manager. The **Add VLB Manager** screen is used to set up communication login credentials to VMware using saved configurations that allow easy communication between Equalizer and VMware.


Add VLB Manager

VLB Manager Name	VLB_Manager_Name
VLB Manager URL	VLB Mgr. URL
Username	touch
Password	****

7. Configure the parameters as described in the "VLB Manager" on page 282 table above.
8. Click on **Commit** to save the new VLB Manager. The new VLB Manager will appear on the right configuration pane list. You can edit it by double-clicking it.

VLB Manager using the CLI

Create a VLB Manager as an External Service as follows:

1. Log in to the CLI.
2. A VLB Manager is a saved configuration by which Equalizer communicates with VMware. Enter the following a the `eqcli` command prompt to create a VLB Manager as an External Service:

```
eqcli > ext_services vlb_manager <name>
```

   where:

   name - is the name of the vlb manager

3. Enter the new VLB Manager, adding a **URL**, **username**, **password**, **Connect Timeout** parameters and flags. Enter:

```
eqcli xs vlb-nam* > URL value
eqcli xs vlb-nam* > username name
eqcli xs vlb-nam* > password name
eqcli xs vlb-nam* > flags disable[a]
```

a. The only flag used is **disable** which would disable the VLB Manager if necessary.

```
eqcli xs vlb-nam* > URL value
eqcli xs vlb-nam* > username name
eqcli xs vlb-nam* > password name
eqcli xs vlb-nam* > flags disable[a]
```

a. The only flag used is **disable** which would disable the VLB Manager if necessary.

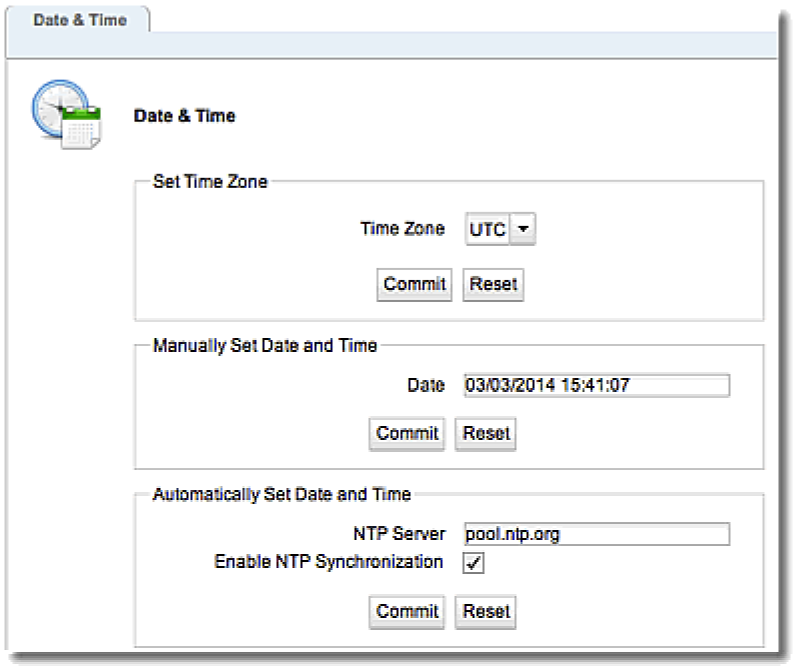
Maintenance

The Maintenance screen (tab) allows you to access the sections in the related topics.

Date and Time

Setting Data and Time using the GUI

The System time setting screen is used to manually enter the current system date and time. This is accessed by selecting Equalizer on the left navigational pane and selecting the **Maintenance** (tab) and then selecting **Date & Time** to display the following.


Date & Time

Date & Time

Set Time Zone

Time Zone: UTC

Commit Reset

Manually Set Date and Time

Date: 03/03/2014 15:41:07

Commit Reset

Automatically Set Date and Time

NTP Server: pool.ntp.org

Enable NTP Synchronization: ✓

Commit Reset

Set Time Zone

Use the Time Zone drop down list to set the Time Zone. Click on **Commit** to save the settings. Click on **Reset** to reset all the newly entered values to the previous values.

Manually Set Date and Time

Enter the current date and time in the **Date** field in the format: `mm/dd/yyyy hh:mm:ss` Click on **Commit** to save the settings.Click on **Reset** to reset all the newly entered values to the previous values.

Automatically Set Date and Time

Network Time Protocol (NTP) is a protocol and software implementation for synchronizing the clocks of computer systems. It provides Coordinated Universal Time (UTC) including scheduled leap second adjustments and will be used to synchronize the clock in Equalizer. Select the **Enable NTP Synchronization** option to enable this feature. Click on **Commit** to save the settings.

Refer to Selecting an NTP Server for additional information about the NTP servers.

Setting Date and Time using the CLI

Refer to "Global Commands" on page 164 to set up date and time using the CLI.

Backup and Restore

The Backup and Restore feature is used to backup your system configuration into a restorable file and provide you with the ability to re-install the cluster/server pool/server configuration, Envoy configuration, networking details, licensing and certificates on your appliance if necessary. For example, if you purchase a new appliance or need a unit to be configured in failover, and you need the same configuration to be quickly implemented on the new system; the Restore feature could set up the same configuration as your current system.

Using the CLI or GUI, a backup file can be generated that contains the following Equalizer information:

- A snapshot of all load balancing objects in the configuration: including clusters, server pools, servers, health checks, responders, failover peers, LLB gateways, GeoClusters and GeoSites, etc.
- All login and authentication information, the VLAN and subnet configuration, logging, remote access, etc.
- Licensing information (if present).
- SSL Certificates used by the GUI. (Cluster SSL certificates are not backed up for security reasons)

The backup file can be uploaded to an FTP site or saved locally.

Using the CLI or GUI, the Restore feature allows you to restore the backup file, with configured objects and parameters to Equalizer. You can transfer the backup file to Equalizer using FTP or by uploading a backup file through the browser. Using FTP requires that you first make the backup file available for download from a local FTP server.

Be aware that when restoring a backup file, the configuration existing on your Equalizer will be overwritten and the system will be rebooted. Your SSL certificates (if any) will be erased and need to be reinstalled. You will be prompted to continue with the restore after a warning is displayed.

You must have Administrator permissions on Equalizer to perform backup/restore.

Backup/Restore and Failover Configurations

When applying a backup file to a replace a unit that is in a failover configuration, it is possible that the peer definitions restored from the backup file may be removed from the running system the next time the system boots. This is because of the way the system ensures the integrity of the local peer information in the configuration file. Each time the system boots, it looks for a local peer definition in the configuration whose System ID number matches the System ID of the hardware on which the system is running. It then does one of the following:

- If a unique local peer definition *is* found, the System ID found in the local peer definition is compared against the System ID being used by the running system. If they do not match (as in the case where a backup file from one Equalizer is being restored on another), the configuration file is modified to reflect the System ID of the running system and the signature is re-generated. If they do match, the configuration is not modified.

- If a unique local peer definition *is not* found, then all peer definitions are removed from the configuration file and a new local peer definition is generated. This behavior is the same behavior that occurs if Equalizeris booted and there are no peer definitions found in the configuration file - such as when the system is reset to factory defaults.

"Failover" on page 683 for additional information.

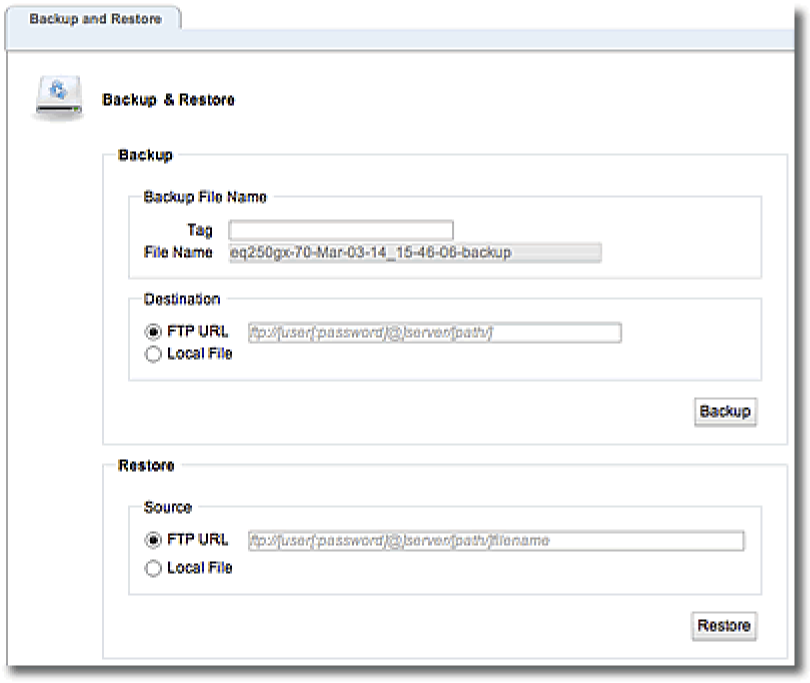
Restore Notes

1. Using the CLI, restore a backup archive from a local directory is not supported.
2. When restoring a backup archive created on an Equalizer other than the one you are restoring, all IP addresses (clusters, servers, failover IP addresses, VLAN IP addresses, etc.) will be instantiated as-is from the backup archive. Consequently, if the unit on which the backup archive was created is connected to the network, IP conflicts will arise. You must correct the IP address conflicts before configuring the restored unit into failover, or issues will occur.
3. If a backup was performed on a system with more interfaces than exist on the system on which it is being restored, full connectivity will not be restored if a VLAN specifies a port that does not exist on the system on which it is being restored. Connectivity can only be restored for VLANs that specify ports that exist on the system on which they are being restored. Contact Support if you continue to see networking issues.
4. Any valid licenses present on the running system are preserved and the licenses in the backup file are discarded. If there are no valid licenses on the running system, any licenses in the backup file are restored.

Backup using the GUI

Follow the steps below to create a backup archive of your current system configuration using the GUI.

1. Log in to the GUI as described in "Logging In" on page 238.
2. Click on the **Maintenance** tab and then select **Backup and Restore.** The following will be displayed.



Backup and Restore

Backup & Restore

Backup

Backup File Name

Tag:

File Name: eq250gx-70-Mar-03-14_15-46-06-backup

Destination

(●) FTP URL: ftp://user[:password]@server[path/]

( ) Local File

Restore

Source

(●) FTP URL: ftp://user[:password]@server[path/]filename

( ) Local File

3. In the **Backup** pane enter a **File Name** which is built from the optionally specified **Tag.** In the example above, **voodoo** is used. The **Tag** is used in addition to the default file name, which is of the format:

```
system_name>[-<tag>]-<date>_<time>-backup.tbz
```

As the **Tag** is typed, it is added to the **File Name** or, at least, when the focus leaves the **Tag**, the tag should be added to the **File Name** - which is read-only.

4. In the **Destination** section, select either **FTP URL** to upload to an FTP site or **Local File** to save the file locally.

   a. For **FTP URL**, you must type the full FTP URL path to the backup file leaving off the file name. A terminating slash (/) is required. The italic text shown indicates the required URL format. Entry of an FTP URL will replace the italic text.

   b. If the **Local File** option is selected when you click on the **Backup** button a save location dialogue box will be displayed. Select a location to save the file and enter a file name. Click **Save** to save the file locally.

5. Click on the **Backup** button to create the backup file either save it to the specified local directory or transfers the file to the FTP server via the URL entered.

Backup using the CLI

The backup archive is created and then uploaded to a URL that specifies an FTP site that can be reached by Equalizer.

To create a backup and upload to a specific URL, enter the following:

```
eqcli > backup url URL  name
```

where:

name is a string that will be used in the backup file name. The default name is of the form:

URL is the path to save the backup and must be in the form:

```
ftp://[user[:password]@]server[/path]
```

The backup file will be in the following format:

```
systemname-date_time-backup.tbz
```

When you restore , the ***name*** is required and must match the name of the backup archive to be downloaded.

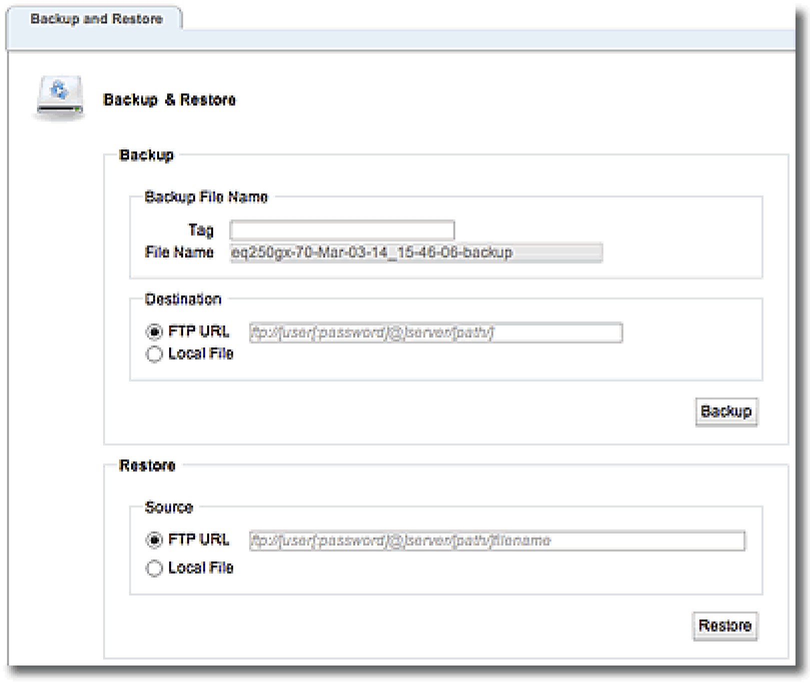
Note - You will be prompted to enter a password if it is not supplied in the URL

Note - If the system you are restoring is not the system on which the backup archive you are using was created, there may be issues to address after the restore is complete and the system reboots. See Restore Notes.

Restore using the GUI

Follow the steps below to restore a backup file containing all user-configured objects and parameters through the GUI:

1. Log in to the GUI as described in "Logging In" on page 238.
2. Click on the **Maintenance** tab and then select **Backup and Restore.** The following will be displayed.



Backup and Restore

Backup & Restore

Backup

Backup File Name

Tag:

File Name: eq250gx-70-Mar-03-14_15-46-06-backup

Destination

(●) FTP URL *ftp://[user[:password]@]server[path/]*

( ) Local File

Backup

Restore

Source

(●) FTP URL *ftp://[user[:password]@]server[path/]filename*

( ) Local File

Restore

3. In the **Restore** section select either **FTP URL** or **Local File.**

   For **FTP URL** you must type in the full path name (including the file name) into the text box. The italic text shown indicates the required format. Entry of an FTP URL will replace the italic text. When the **Restore** button is clicked, the file is downloaded from the specified FTP site and a pop up displays a summary of the configuration in the archive. A notification SSL Certificates for HTTPS cluster notification will be displayed. Click on **Continue** or **Cancel**. If **Continue** is selected the archive is restored and the system is rebooted.

   For **Local File** click on the **Restore** button to display a file selection dialogue to select a file from local storage. Once the file is selected, it is transferred to Equalizer and a popup displays a summary of the configuration in the archive. A notification SSL Certificates for HTTPS cluster notification will be displayed. Click on **Continue** or **Cancel**. If **Continue** is selected the archive is restored and the system is rebooted.

Restore using the CLI

The previously archived backup is uploaded from a URL that specifies an FTP site that can be reached by Equalizer.

To restore a previously backed up file from a specified URL (location) enter the following:

```
eqcli > restore url URL name
```

Where:

name must match the name of the backup archive to be downloaded.

URL is the path to the previously backed up file and must be of the form:

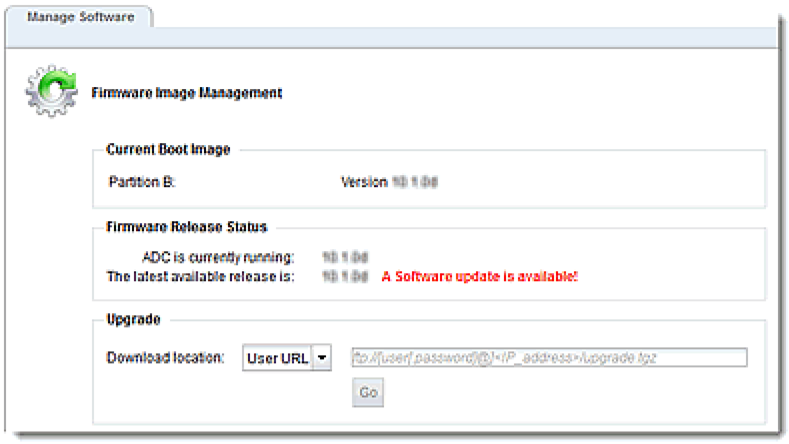
ftp://[user[:password]@]server[/path]

Note - You will be prompted to enter a password if it is not supplied in the URL

Manage Software

You can upgrade your version of the operating system using the **Manage Software** screen on the GUI.

1. Click on the **System** configuration tab.
2. Click on the arrow (▶) beside **Maintenance** to expand the branch.
3. Select **Manage Software** to display the following.



Manage Software

Firmware Image Management

Current Boot Image

Partition B: Version [illegible]

Firmware Release Status

ADC is currently running: [illegible]

The latest available release is: [illegible] A Software update is available!

Upgrade

Download location: User URL ftp://[user[:password]@]<IP_address>/upgrade.tgz

Go

Current Boot Image

The current boot image and the partition where it resides is displayed.

Firmware Release Status

If the release running on your EQ/OS is older than the latest version available, then the red bold text with a message *"A software update is available"* will be displayed. If your browser cannot download the well-known page, then the Latest available release is indication will display "Connection to CPS website unsuccessful." and a URL to the download site will be displayed in the **CPS download URL space**.

Upgrade

To download and install an image, select the **Download location** using the drop down list.

If **Local File** is selected you will be prompted for the location of the local file. When a file is selected the path and file name will be displayed.

If **CPS URL** is selected the CPS download URL pointing to **the latest available release** shown in the **EQ/OS Release Status** pane will automatically be displayed on the right.

If **User URL** is selected enter a URL in the text box on the right that points to a single-file download image. The file path should be in the format: *ftp://[user[:password]@]<IP_ address/upgrade.tgz>*

After entering the **Download location** click on **Go** to begin the upgrade. If successful a *"GUI Installation successfully completed"* message will be displayed.

If the upgrade does not complete successfully, verify your network connectivity and the location of the upgrade file and attempt the upgrade again.

Tools

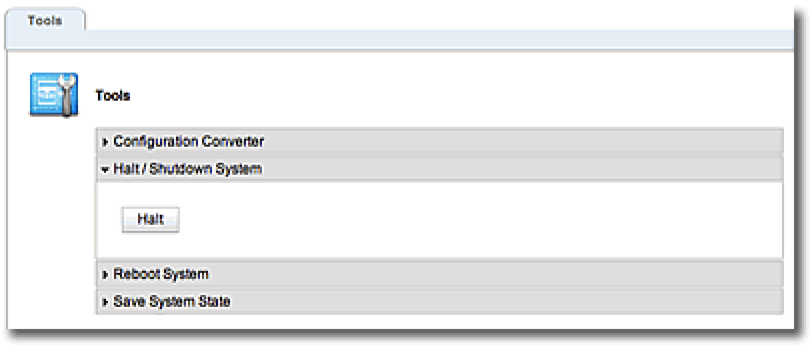
The Tools screen provides useful utilities that includes:

- A **Configuration Converter** that allows you to create an EQ/OS 10 configuration that is functionally equivalent to the EQ/OS 8.6 configuration from a supplied backup archive.
- A **Halt/Shutdown** command, allows you to turn your Equalizer "off" from directly in the GUI.
- A **Reboot System** command, allows you to reboot your Equalizer from directly in the GUI.
- A **Save System State** feature, that allows you to create an archive of your various configuration files, logs and other details used to help in diagnosing any issues that may arise.

Access any of the tools by selecting the appropriate accordion tab to display the commands/details.

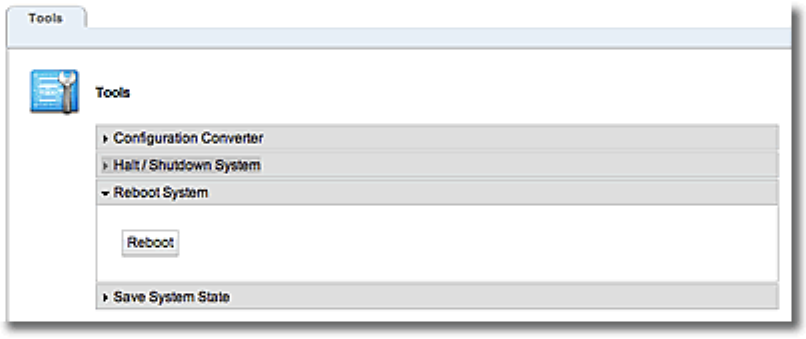
Halt/Shutdown System

Click on the **Halt/Shutdown System** configuration tab to display the following. Click on the **Halt** button to shut down your Equalizer.



Reboot System

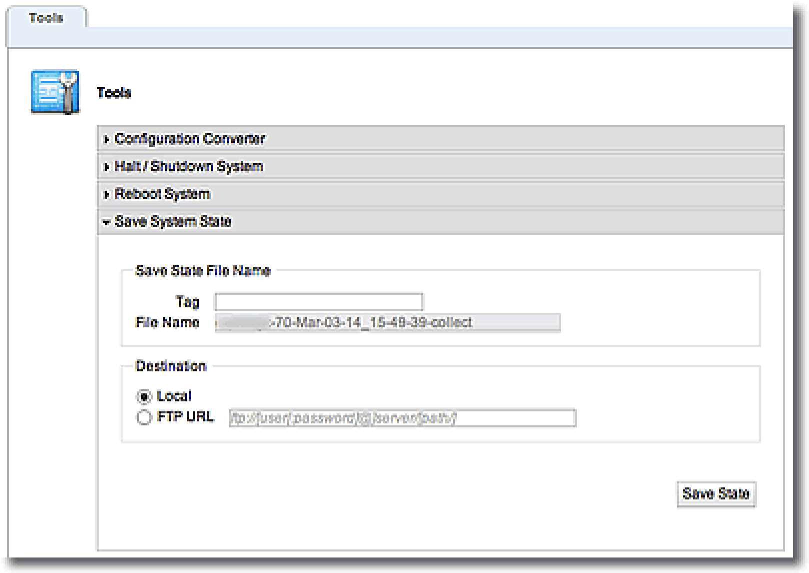
Click on the Reboot System configuration tab to display the following. Click on the **Reboot** button to reboot your Equalizer.



Save System State

Click on the **Save System State** configuration tab to display the following. In this screen you can set up a Save State or system information archive that contains various configuration files, logs, and other information used by Support to help diagnose problems you are having with Equalizer. The file can be saved locally or uploaded to an FTP server.

1. Click on the **Maintenance** tab and then **Save State** to display the following:



2. Enter a **File Name** for the archive. This should be in the format: ***`hostname-tag-date_time-`*** **`collect.tbz`** If desired, enter a unique **`tag`** for the archive file name in the **Tag** field which will be included in the archive file name. If a tag is not entered, the file name will be in the format specified, however, without that **`tag`** element.

3. Select either the **Local** or **FTPURL** option in the **Destination** pane.
   a. If you select **Local**, the archive will be saved in the default “save” directory specified in your web browser options.
   b. If you select **FTP URL**, enter the URL of the FTP site on which you will upload the archive file.The URL should be in the format: ***`ftp://[user[:password]@] server/[path/].`***
4. Select the **Save State** button to create the archive. Once Equalizer collects the information for the archive, a dialog box is displayed by your browser to open or save the archive.
5. If you require technical support, open your email client and send the file you saved to **support@coyotepoint.com** as an attachment or provide the URL (with credentials) for the FTP site on which the archive now resides. Explain the nature of your problem in the email, or just include the support ticket number you were given previously by Coyote Point Support.

Network Configuration

Clicking on the **System** configuration tab on the GUI and then the ▶ beside Network will expand the branch to provide access to the **Interface** configuration screen, the **Link Aggregation** configuration screen, the **VLAN** configuration screen and the IPv6 **Tunnel** configuration screen.

Interfaces

To view statistics for an interface in the GUI, select the **System** configuration tab and then click on the arrow (▸) beside **Network** to expand the branch. Select **Interface** and then click on a port on the Equalizer port display on the right to display statistics.

Refer to "Interface Commands" on page 195 for a complete listing of CLI Interface commands

Full	If you select this option the load balancer will attempt to auto negotiate the highest available speed with the unit on the other end of the connection.
Duplex Mode	If the port status is Link Up, this is the current port duplex setting. If the status is Link Down, this is either the highest duplex that can be negotiated, or the force setting. Can be set to **Full** or **Half**.
Speed	If the **Port Status** is **Link Up**, this is the current port speed. If the **Port Status** is **Link Down**, this is the highest speed that can be negotiated, or the force setting. Can be set to **10**, **100**, or **1000** Mbps. On the E370LX Equalizer, the link light will be displayed in amber ● when connected at 1 Gbps and green ● when connected at 100 Mbps. On the E470LX, E670LX and E970LX Equalizers link light on Equalizer's ports will be displayed in green ● when connected at 1 Gbps and amber ● when connected at 100 Mbps. In addition, ports 9 and 10 on Equalizer E670LX and E970LX will be displayed in green ● when connected at 10 Gbps. The 10Gb interface ports on E670LX and E970LX will always attempt to negotiate 10Gb with whatever device is on the other end of the wire, and will not negotiate lower speeds.

Note -The following 10Gb SFPs (Small Form-factor Pluggable modules) are supported:

(Equalizer E670LX and E970LX only)
10GbaseLR - single-mode fiber
10GBase-SR 850nm Multi-mode
10GBase CX4 copper
10GBase Twinax copper
10GBase Twinax Long copper
10GBase-LRM 850nm Multi-mode
10GBase-T - RJ45

If you would like to display statistics using the CLI,enter the following:

```
eqcli > interface interface-name stats
```

The tables below show a typical port statistics displays.

Transmitted Counters	
Packets	The total number of transmitted packets on this interface.
bytes	The total number of bytes transmitted on this interface
multicasts	The total number of good broadcast/multicast (e.g., ARP) packets transmitted by this interface.
errors	The total number of bad packets transmitted by this interface.
collisions	The total number of packets that were dropped (e.g., lack of transmit buffer , collision detection). These packets are not transmitted by the interface.
Received Counters	
Packets	The total number of packets received on this interface.
bytes	The total number of bytes received on this interface.
multicasts	The total number of good broadcast/multicast (e.g., ARP) packets received on this interface.
errors	The total number of bad packets (e.g., CRC errors,, alignment errors) received on this interface.
drops	The total number of packets that were dropped (e.g., lack of receive buffer, congestion, invalid classification, e.g., tagged frame received on untagged port) by the receiving interface.
unknown protocol	Tot total number of packets received on this interface that used an unknown protocol.

Link Aggregation

Note -Link Aggregation is supported on the LX series, all virtual platforms, and on legacy E250GX systems *only*.

Link aggregation combines multiple physical interfaces into a single aggregated (or, logical) interface, providing increased bandwidth as well as link redundancy. Traffic is distributed evenly over the physical links of the aggregation group; and, if one of the links in the aggregated interface becomes unavailable, traffic will continue to flow over the available interfaces in the group.

Link Aggregation Protocol (LACP)

LACP is a protocol used between network devices to automatically bundle links between the devices, and is supported by link aggregation. Once you configure an aggregated interface with LACP enabled, LACP packets are broadcast to other directly connected devices (such as switches and routers), which will create the necessary aggregated links (if they are also enabled for LACP).

Aggregated links on other network devices must be manually created on those devices if either LACP is disabled on the aggregated interface you create, or if a network device does not support LACP.

LACP supports active mode only; passive mode LACP is not supported.

General Process for Creating Aggregated Interfaces

1. Configuration of aggregated interfaces via the CLI/GUI by specifying:

    a. A unique aggregated interface name.

    b. The physical interfaces (ports) to be configured as members of the aggregated interface.

    c. A flag indicating whether LACP is to be enabled or disabled (it is enabled by default).

2. Assign the aggregated interface to a VLAN by adding an interface instance of the aggregation group to the VLAN

Limitations

1. A maximum of 4 physical interfaces may be combined into one aggregated interface.
2. A physical interface may belong to no more than 1 aggregated interface.
3. An aggregated interface may be specified as an untagged interface in no more than one VLAN. (There are no limitations for aggregated interfaces used as tagged interfaces; in other words, an aggregated interface may be specified as a tagged interface in multiple VLANs).
4. When assigning interfaces (physical or aggregated) to a VLAN, only one interface (physical or aggregated) can be assigned to a VLAN. In other words, if you want to assign two physical interfaces to the same VLAN, you must first create an aggregated interface containing those two physical interfaces, and then assign the aggregated interface to the VLAN.

Configuring Link Aggregation using the CLI

To create a link aggregation group and assign it to a VLAN using the CLI, do the following:

1. Create an aggregation group as follows.

```
eqcli > agr agr00
```

2. Specify the physical interfaces to be added to the aggregation group:

```
eqcli > agr agr00 ifi ge01,ge02
```

   In the command line above, **`agr00`** is the name assigned to the aggregation group; **`ge01`** and **`ge01`** are the physical interfaces (front panel ports) assigned as interfaces instances (**`ifi`**) to the group.

3. Verify the aggregation configuration by entering:

```
eqcli > show agr agr00
AGR Name       : agr00
Flags          : lacp
Interfaces     : ge01, ge02
ge01 Partner   : 08:9e:01:ba:51:61
ge02 Partner   : 08:9e:01:ba:51:61
eqcli >
```

   The status of each physical interface in the interface instance is displayed as a "`Partner`" in the aggregation summary display.(e.g., `ge01 Partner`, `ge02 Partner`).

   - When link aggregation is enabled, a MAC address of a "partner" is reported in the show display The MAC address will be that of the first member of the aggregated link on the "partner" side.
   - If no link is established, `Not Present` will be displayed. This indicates that the other (partner) side of the aggregated interface is not configured or that it is configured as "static". When both sides configure the aggregated link as dynamic, link aggregation is enabled. This means that MAC addresses are exchanged.
   - When "Link down" is reported as the "partner" status there is a problem connecting to a port on the partner side of the aggregated link.

Note - The lacp flag is enabled by default.

4. If you have not done so already, create a VLAN and do not assign any interfaces to it. See "Configuring VLANs" on page 304 for how to create a VLAN.

5. Add an instance of the new link aggregation group created above. In the example below, an interface instance of the **agr00** aggregated link created above is added to the VLAN **v100** as a tagged interface.

```
eqcli > vlan v100 ifi agr00 type tagged
```

6. Verify the new VLAN configuration and associated interfaces by entering:

```
eqcli > show vlan v100

VLAN Name        : v100
VID              : 1
MTU              : 1500
Subnets          : sn02
Interfaces       : agr00
```

Removing an Aggregated Interface from a VLAN using the CLI

To remove an aggregated interface from the VLAN used in the previous example above, enter the following command:

```
eqcli > no vlan v100 ifi agrname
```

Removing an Aggregated Interface from the System using the CLI

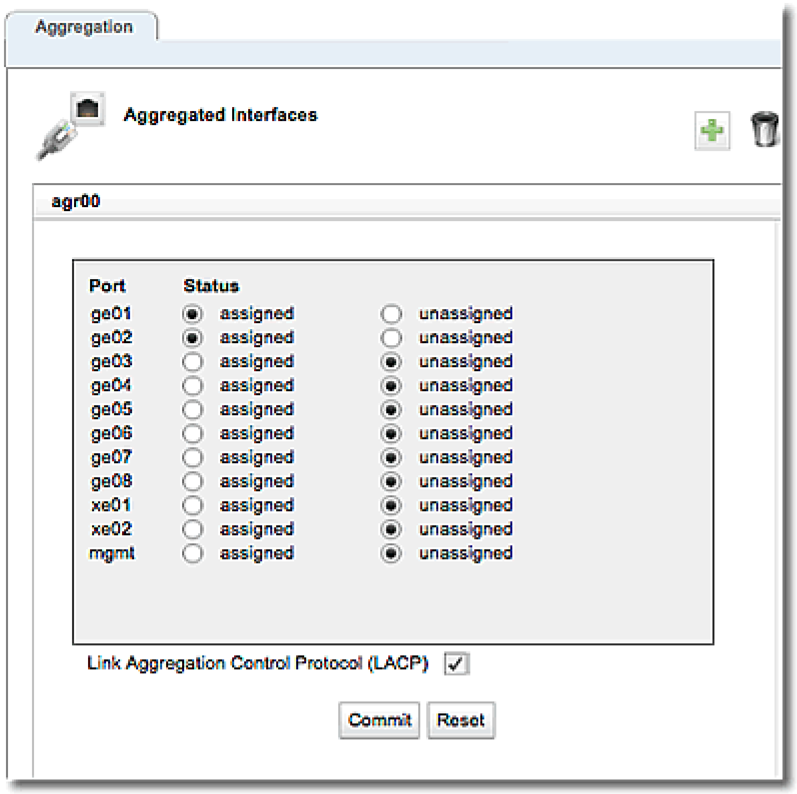
To delete an aggregated interface from the system, you should first remove it from all VLANs that use it (see above), and then enter the following command:

```
eqcli > no agr agrname
```

Configuring Link Aggregation using the GUI

To create a link aggregation group and assign it to a VLAN using the GUI, do the following:

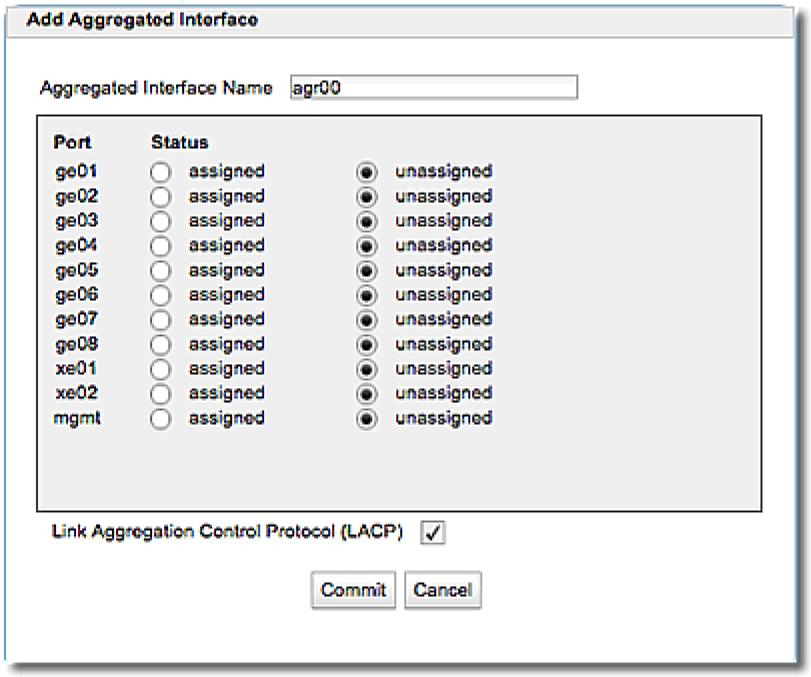
1. Click on the **System** configuration tab on the left pane of the GUI if it is not already selected.
2. Click on the arrow (▶) beside **Network** to expand the branch.
3. Click on **Aggregation** to display the Aggregation configuration screen. The screen features accordian tabs that will display the configured **Aggregated Interfaces** when clicked. The example below shows E670LX/E970LX.



Port	Status	
ge01	[✓ assigned]	unassigned
ge02	[✓ assigned]	unassigned
ge03	assigned	[✓ unassigned]
ge04	assigned	[✓ unassigned]
ge05	assigned	[✓ unassigned]
ge06	assigned	[✓ unassigned]
ge07	assigned	[✓ unassigned]
ge08	assigned	[✓ unassigned]
xe01	assigned	[✓ unassigned]
xe02	assigned	[✓ unassigned]
mgmt	assigned	[✓ unassigned]

Link Aggregation Control Protocol (LACP) ✓

4. Click on ➕ on the **Aggregated Interfaces** configuration screen. The **Add Aggregated Interfaces** dialogue shown below will be displayed. (E670LX/E970LX shown)



Add Aggregated Interface

Aggregated Interface Name: agr00

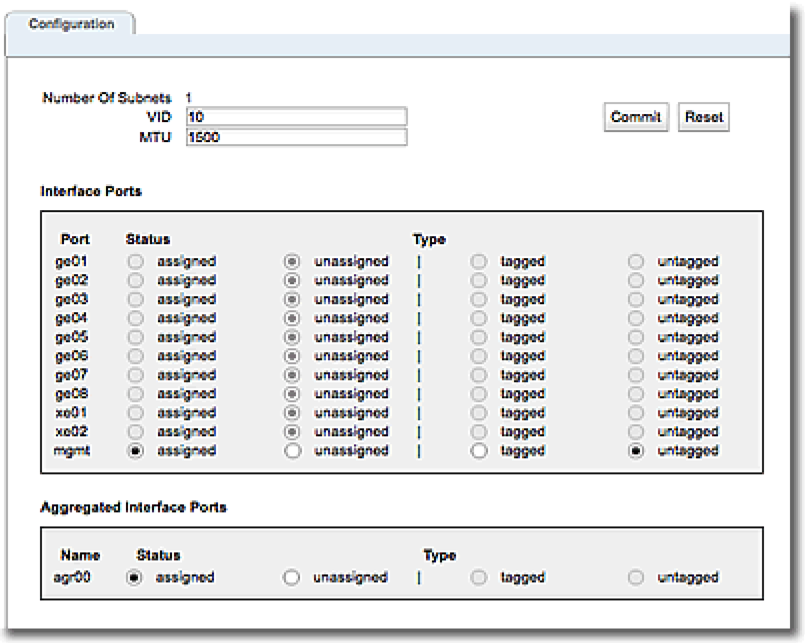
Port	Status	
ge01	○ assigned	● unassigned
ge02	○ assigned	● unassigned
ge03	○ assigned	● unassigned
ge04	○ assigned	● unassigned
ge05	○ assigned	● unassigned
ge06	○ assigned	● unassigned
ge07	○ assigned	● unassigned
ge08	○ assigned	● unassigned
xe01	○ assigned	● unassigned
xe02	○ assigned	● unassigned
mgmt	○ assigned	● unassigned

Link Aggregation Control Protocol (LACP) ✓

5. Enter a name in the **Aggregated Interface Name** field and select the **assigned** radio button on ports to be included in the aggregated interface.

Note - The Link Aggregation Control Protocol (LACP) flag is enabled by default.

6. Click on **Commit** to save the new aggregated interface.
7. Click on a VLAN on the left navigational pane and the aggregated interfaces that have been created will be displayed as shown below on the **VLAN Configuration** screen. Add **Aggregated interfaces** to the VLAN by selecting radio buttons . VLANs can either be **tagged** or **untagged**. **tagged** ports can be assigned to more than one VLAN. **untagged** interfaces can be assigned to exactly one VLAN. Select an appropriate radio button in **Type**. (E670LX/E970LX shown)



Configuration

Number Of Subnets 1
VID 10
MTU 1500

Commit Reset

Interface Ports

Port	Status		Type	
ge01	assigned	[✓ unassigned]	tagged	untagged
ge02	assigned	[✓ unassigned]	tagged	untagged
ge03	assigned	[✓ unassigned]	tagged	untagged
ge04	assigned	[✓ unassigned]	tagged	untagged
ge05	assigned	[✓ unassigned]	tagged	untagged
ge06	assigned	[✓ unassigned]	tagged	untagged
ge07	assigned	[✓ unassigned]	tagged	untagged
ge08	assigned	[✓ unassigned]	tagged	untagged
xe01	assigned	[✓ unassigned]	tagged	untagged
xe02	assigned	[✓ unassigned]	tagged	untagged
mgmt	[✓ assigned]	unassigned	tagged	[✓ untagged]

Aggregated Interface Ports

Name	Status		Type	
agr00	[✓ assigned]	unassigned	tagged	untagged

8. Click on **Commit** to save assignments.

Removing an Aggregated Interface from a VLAN using the GUI

To remove an aggregated interface from a VLAN:

1. Select a VLAN from the left navigational pane to display the VLAN **Configuration** screen.
2. Select the unassigned radio button from the **Aggregated Interface**s pane at the bottom of the screen.
3. Click on **Commit** to complete the process.

Removing an Aggregated Interface from the system using the GUI

1. Click on the **System** configuration tab on the left pane of the GUI if it is not already selected.
2. Click on the arrow (▶) beside **Network** to expand the branch.
3. Click on **Aggregation** to display the **Aggregated Interfaces** screen.
4. Select the **Aggregated Interfaces** accordian tab that you want to remove from the system.
5. Click on to remove the interface.
6. Click on **Commit** to complete the process.

Configuring VLANs

The following table shows you how to perform VLAN tasks using the CLI and the GUI:

It should be noted that on switch less system only one port can be assigned to a VLAN. On Equalizers with a front-panel switch (E350GX, E450GX, E650GX), multiple ports can be assigned to a VLAN.

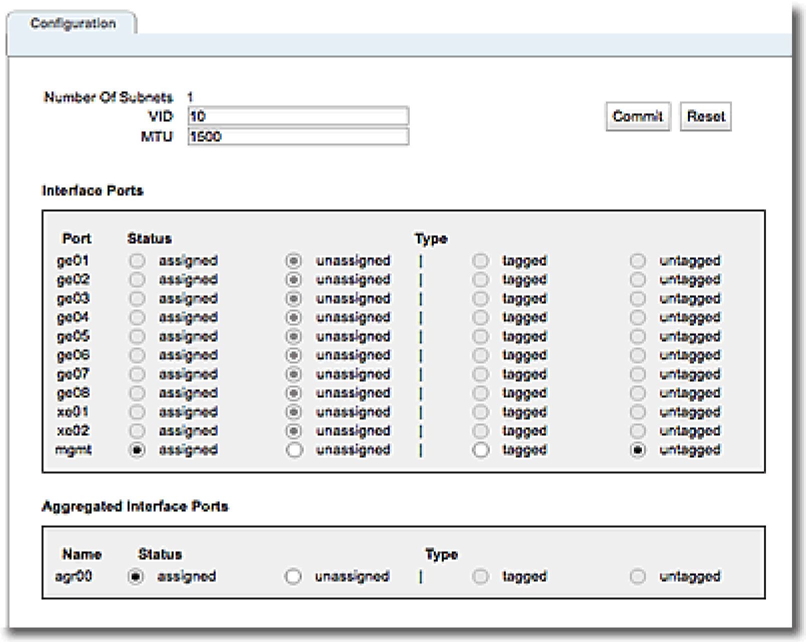
Note - The **VID** values must be between 1 and 4094.

CLI and GUI VLAN Commands

Task	Command / Procedure	
Add a VLAN	**CLI**	`eqcli > vlan name vid VLAN_ID [parameters]`
	GUI	1. Right-click **VLANs** in the left frame. 2. Select **Add VLAN** from the popup command menu. 3. Enter a **VLAN Name** and **VID** (VLAN ID). 4. Click **Commit**.
Remove a VLAN	**CLI**	`eqcli > no vlan name`
	GUI	1. Expand the **VLANs** node in the left frame. 2. Right-click the name of the VLAN you want to delete. 3. Select **Delete VLAN** from the pop up command menu. 4. Click **Confirm**.
Modify a VLAN	**CLI**	`eqcli > vlan name [parameters]`
	GUI	1. Expand the **VLANs** node in the left frame. 2. Click the name of the VLAN you want to modify. The VLAN configuration tabs appear in the right frame. 3. Edit the VLAN configuration using the controls on each tab. Click **Commit** before navigating away from a tab to apply your changes.
Display summary of all VLANs	**CLI**	`eqcli > show vlan`
	GUI	Click the **VLANs** node in the left frame object tree.
Display details for a VLAN	**CLI**	`eqcli > show vlan name`
	GUI	1. Expand the **VLANs** node in the left frame. 2. Click a VLAN name to display the configuration tabs for that VLAN in the right frame.

VLAN Port Assignment Using the GUI

The VLAN Port Assignment Configuration Screen is used to assign ports, specify whether a VLAN is tagged or untagged and specify MTU. It is accessed by clicking on a VLAN on the left navigational pane of on the GUI. Consider the following example:



Number Of Subnets 1
VID 10
MTU 1500

Interface Ports

Port	Status		Type	
ge01	○ assigned	● unassigned	○ tagged	○ untagged
ge02	○ assigned	● unassigned	○ tagged	○ untagged
ge03	○ assigned	● unassigned	○ tagged	○ untagged
ge04	○ assigned	● unassigned	○ tagged	○ untagged
ge05	○ assigned	● unassigned	○ tagged	○ untagged
ge06	○ assigned	● unassigned	○ tagged	○ untagged
ge07	○ assigned	● unassigned	○ tagged	○ untagged
ge08	○ assigned	● unassigned	○ tagged	○ untagged
xe01	○ assigned	● unassigned	○ tagged	○ untagged
xe02	○ assigned	● unassigned	○ tagged	○ untagged
mgmt	● assigned	○ unassigned	○ tagged	● untagged

Aggregated Interface Ports

Name	Status		Type	
agr00	● assigned	○ unassigned	○ tagged	○ untagged

The GUI displays an icon in the left navigational pane next to the name of any VLAN if it does not have any assigned interfaces. Moving the mouse over the icon displays text indicating that the VLAN configuration is incomplete.

The VLAN Port configuration on the E670LX and E970LX displays

- 8-1GB ports
- 1-1GB management port
- 2-10GB ports and.as many aggregated ports as are currently configured by the user.

The E470LX and E370LX do not have 10GB ports so the 10 GB pane is not displayed on the VLAN Port configuration screen.

The E370LX does not have a MGMT port and does not appear in the 1GB pane.

On the LX Series of Equalizers, only 1 port can be assigned to a VLAN. Aggregated ports, however, can be configured. Refer to "Link Aggregation" on page 298for details.

VID	A unique integer identifier for the VLAN, between 1 and 4094.

MTU	MTU can be specified for tagged and untagged VLANs on all switched systems (E350GX, E450GX, E650GX)for tagged VLANs on non-switched systems (E250GX,LX Series, Equalizer OnDemand. The MTU is set on the VLAN, and the values you can set depend on the Equalizer model and the subnet configuration of the VLAN, as follows: For the E350GX, E450GX, E650GX, and E370LX, the maximum MTU value is 4839. For E250GX models and Equalizer OnDemand, the maximum MTU is 9000. For VLANs with only IPv4 subnets, the minimum MTU is 576. For VLANs with an IPv6 subnet, the minimum MTU is 1280. If you modify the MTU on a VLAN to a value that is lower than the currently set value, you must reboot Equalizer to ensure proper network interface operation.
Status	Assign a port status on the VLAN using the radio buttons.
Type	VLANs can either be **tagged** or **untagged** and set for the port on the VLAN using the appropriate radio button: **Tagged** ports can be assigned to more than one VLAN. **Untagged** ports can be assigned to exactly one VLAN.

Click on **Commit** to save your settings or **Reset** to revert to the previous settings.

VLAN Port Assignment Using the CLI

Configure a VLAN using the CLI as follows:

Create a VLAN and subnet over which you can log into Equalizer. If you want to license Equalizeronline, the subnet you create should also be able to reach the Internet.

1. To create a VLAN , enter a command like the following:

```
eqcli > vlan vlname vid vlan_ID
```

   Replace ***vlname*** with the VLAN name and **vlan_ID** with the VLAN ID number (1-4094). If you are using untagged VLANs (common in many sites), the VLAN ID can be any number not used on another VLAN. If you are using tagged VLANs, check with your network Administrator for the correct VLAN ID to specify.

2. Add a subnet to the VLAN you just created. You’ll need to specify the following:

- The subnet IP address, which is the ADCs address on this network. It must be an IPv4 or IPv6 address in CIDR (Classless Inter-Domain Routing) format (e.g., 172.16.0.123/21).
- The default route IP address for the subnet gateway. This is an unadorned IP address (e.g., 172.16.0.1).
- The HTTP and SSH services, so that you can log in to the Graphical User Interface (GUI) and Command Line Interface (CLI) on this subnet.

Enter the following command, all on one line:

```
eqcli > vlan vlname subnet subnet name ip CIDR format IP address route dest_
cidr src src cidr gw ip_addr
```

In the command above,***vlname*** is the VLAN name from the previous step, ***subnet name*** is the name of the subnet, ***ip***is the CIDR format IP address, ***route***is the destination network in CIDR notation, **src** is the source network in CIDR notation (optional), and **gw** is IP address of the gateway for the route.

Refer to the webhelp if you need more help setting up your initial VLAN and subnet: go to www.coyotepoint.com, move your mouse over the **Support** link near the top of the screen, and choose **Manuals** from the drop down list.

3. Associate an interface instance with the VLAN. Here we assume that you are using the port labelled '1' on the front panel. Enter one of the following commands, depending on whether the VLAN you created above is untagged or tagged (ask your network administrator if you are unsure):

```
eqcli > vlan vlname ifi ge01 type untagged
eqcli > vlan vlname ifi ge01 type tagged
```

4. Connect the port or ports you configured on the VLAN to the network using a standard Ethernet cable with RJ-45 connectors. You should now be able to use the "ping" command from a workstation on the same subnet to reach the subnet IP address configured above.

Configuring Subnets

The following table describes how to perform subnet tasks using the CLI and the GUI:

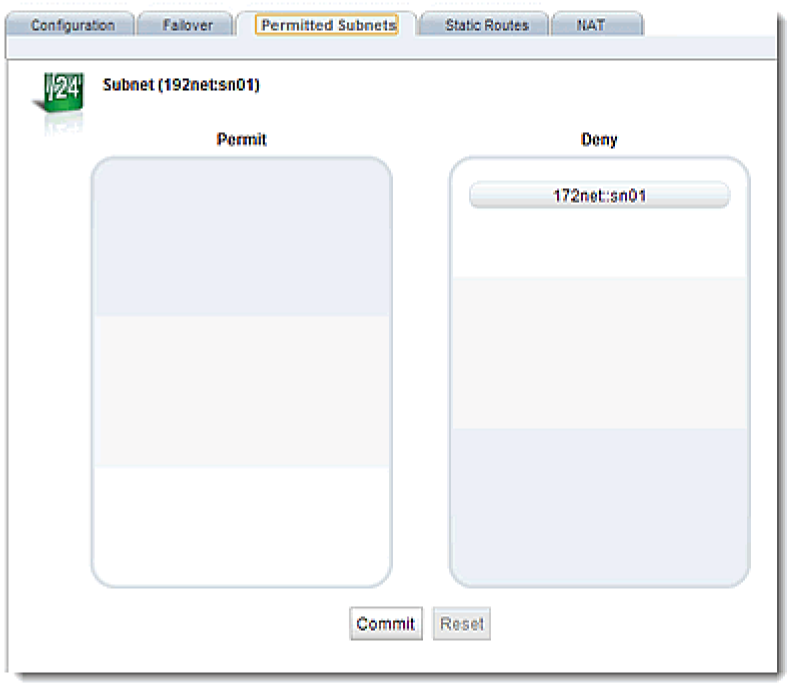
Task		Command / Procedure
Add a subnet	**CLI**	`eqcli > vlan name subnet name [parameters]`
	GUI	1. Click on the **System** configuration tab on the left pane. 2. Click on the arrow (▶) next to **Network** to expand the branch. 3. Click on the arrow (▶) next to **VLANs** to expand the branch to display all configured VLANs 4. Right-click a VLAN name 5. Select **Add Subnet** from the pop up command menu. 6. Enter a **Subnet Name**, **IP Address** and **Default Route** in the pop up window on the right. 7. Click **Commit**.
Remove a subnet	CLI	`eqcli > no vlan name subnet name`
	GUI	1. Click on the **System** configuration tab on the left pane. 2. Click on the arrow (▶) next to **Network** to expand the branch. 3. Click on the arrow (▶) next to **VLANs** to expand the branch to display all configured VLANs 4.Click on the arrow (▶) next to a specific VLAN to expand the branch to display all configured subnets . 5. Right-click the name of the subnet you want to delete. 6. Select **Delete Subnet** from the popup command menu. 7. Click **Confirm**.
Modify a subnet	**CLI**	`eqcli > vlan name subnet name [parameters]`
	GUI	1. Click on the **System** configuration tab on the left pane. 2. Click on the arrow (▶) next to **Network** to expand the branch. 3. Click on the arrow (▶) next to **VLANs** to expand the branch to display all configured VLANs 4.Click on the arrow (▶) next to a specific VLAN to expand the branch to display all configured subnets . 4. Click the name of the subnet you want to modify. The configuration displays appear in the right pane. 5. Edit the subnet configuration using the controls on each tab. 6. Click **Commit** before navigating away to apply your changes.
Display summary of all subnets in a VLAN	**CLI**	`eqcli > show vlan name subnet`
	GUI	N/A

Task	Command / Procedure	
Display details for a subnet	**CLI**	`eqcli > show vlan` *`name`* `subnet` *`name`*
	GUI	1. Click on the **System** configuration tab on the left pane. 2. Click on the arrow (▶) next to **Network** to expand the branch. 3. Click on the arrow (▶) next to **VLANs** to expand the branch to display all configured VLANs 4.Click on the arrow (▶) next to a specific VLAN to expand the branch to display all configured subnets . 5. Click a subnet name to display the configuration displays for that subnet in the right pane.

About Permitted Subnets

By default, each VLAN will not forward packets for any other subnet unless they are specifically designated in the **Permitted Subnets** screen-- sometimes referred to as a "subnet access control list". When a new subnet is added it will be automatically be added to the **Deny** pane on this screen.

If you would like to allow packets from any other subnet to be forwarded they must be added to the **Permit** pane on this screen. Using drag and drop functionality, drag a **Subnet** from the **Deny** pane and drop it in the **Permit** pane to allow packet forwarding on the subnet. Similarly, if you would like to remove a subnet from the **Permit** pane, you can drag and drop to the **Deny** pane.



Click on **Reset** to revert to the default permissions. Click on **Commit** to save any subnet permission changes made.

See "VLAN and Subnet Commands" on page 231 for commands used in permitted subnets using the CLI.

Configuring Subnet Destination Routes

Subnet *destination routes* (also commonly called "static routes") are commonly used to specify routes to destination IP addresses via gateways other than the subnet's default route (also called a *default gateway*). They are called destination routes, since they are used to make routing decisions based on a packet's destination IP address.

default route can be specified for every subnet (previous releases supported a single global default route). If you need to access systems on another subnet that cannot be reached via this gateway, then you need to specify a static route to those systems through the gateway for that subnet.

subnet also has its own subnet-specific static route table. Subnet static routes can be specified via the CLI or the GUI.

Also refer to "Source Based Routing Scenarios" on page 117 for a description of source-based routing scenarios.

Configuring Subnet Static Routes using the GUI

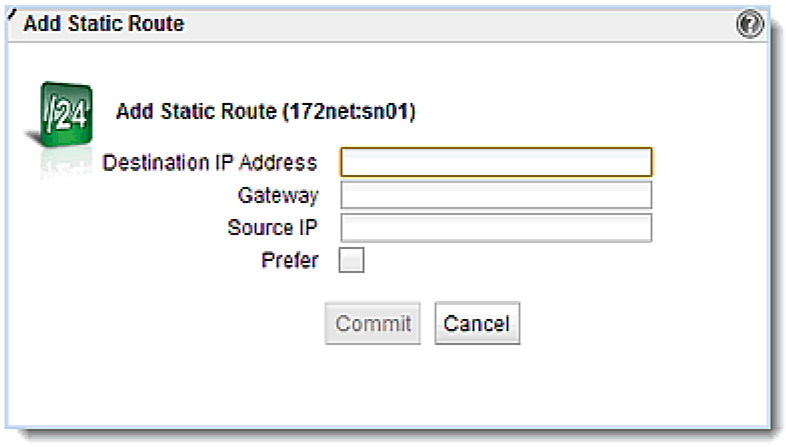
Do one of the following:

1. In the GUI, click on the **System** configuration tab if it is not already selected. Then click on the arrow (▶) beside **VLANs** to expand the branch. Click on the arrow (▶)beside a VLAN to expand the branch to display the subnets. Right-click on the name of a subnet and select **Add Static Route.**

   or

2. In the GUI, click on the **System** configuration tab if it is not already selected. Then click on the arrow (▶) beside **VLANs** to expand the branch. Click on the arrow (▶)beside a VLAN to expand the branch to display the subnets and select a subnet.

   The following will be displayed.



Specify the **Destination IP Address**, **Gateway**, and Source IP address. This determines the route to be used. For example, you may want a set of routes on a subnet such as the following:

1. **Destination IP Address**: 0/0, **Gateway**: 172.16.0.1, **Source IP**: 188.161.0.118.32
2. **Destination IP Address**: 0/0, **Gateway**: 192.168.0., **Source IP**: sn (subnet)

This routing scenario says:If a packet destined to "anywhere" (**Destination IP Address** 0/0) is from **Source IP** 188.161.0.118/32, use **Gateway** 172.16.0.1. Otherwise, use **Gateway** 192.168.0.1.

Destination IP Address	The IP address for the host or subnet. For IPv4, specified as a Classless Internet Domain Routing (CIDR) address (e.g. 192.168.1.0/24). For IPv6, specified using IPv6 subnet notation.
Gateway	The IP address of the gateway used to reach the host or subnet.
Source IP	The IP address of where a packet originates.
Prefer	Enabling this flag allows you to specify the "preferred" route to be used for any matching destination - even if the destination address is on a subnet that is defined on Equalizer. One **Prefer** flag is allowed for each subnet is allowed at this time, however, this can be enabled on as many static routes as necessary.

To modify a static route, highlight the route in the table and click the **Modify** icon . Change the parameters and click **commit.**

To delete a static route, highlight the route in the table and click the **Delete** icon .

Configuring Subnet Static Routes using the CLI

Static routes are specified in the subnet context (See "VLAN and Subnet Commands" on page 231).

To display the static route for a subnet, use the **show** command:

```
eqcli > show vlan vlan_name subnet subnet_name
```

To add a static route from the global context, enter:

```
eqcli > vlan vlan_name subnet subnet_name dest destination IP src source IP gw
gateway IP
```

The parameters are explained as follows:

vlan_name	The name of the VLAN.
subnet_name	The name of the subnet.
dest	The IP address for the host or subnet. For IPv4, specified as a Classless Internet Domain Routing (CIDR) address (e.g. 192.168.1.0/24). For IPv6, specified using IPv6 subnet notation.
gw	The IP address of the gateway used to reach the host or subnet.
src	The IP address of where a packet originates.
Prefer	Enabling this flag allows you to specify the "preferred" route to be used for any matching destination - even if the destination address is on a subnet that is defined on Equalizer. One **Prefer** flag is allowed for each subnet is allowed at this time, however, this can be enabled on as many static routes as necessary.

Note - the "prefer" flag, which allows you to specify the "preferred" route to be used for any matching destination - even if the destination address is on a subnet that is defined on Equalizer- is not available using eqcli at this time.

To delete a static route, use the **no** form of the **route** command.

Configuring Outbound NAT

Enabling outbound NAT allows servers on a non-routable network to communicate with hosts on the internet by mapping the server's IP address to another IP address that is routable on the internet. On Equalizer, this is disabled by default. Enabling this option has a performance impact, since Equalizer needs to modify every packet sent and received on server subnets.

Outbound NAT can be configured to map a server's IP address to any IP address in the IP address range defined for that subnet. The IP address can be either:

- **An already instantiated IP address on that subnet for a system object, such as:**
  - **the subnet IP address**
  - **the failover IP address**
  - **any cluster IP address**
- **An IP address that is not already instantiated on that subnet.**

If the NAT address that you enter is not already instantiated on the outbound subnet, it will be instantiated. ***Therefore, you need to make sure that no other device on the subnet is using the NAT IP address you specify, or there will be duplicate IP addresses on the subnet that will cause connectivity issues.***

Note - Because outbound NAT is configured on a subnet basis, individual servers cannot be set up for different outbound NAT IP addresses unless they are in different subnets.

When outbound NAT rules are configured for a subnet, the system treats packets on that subnet as if they are part of the external subnet through which they are being NAT'd.

Configuring outbound NAT using the GUI

1. Log into the GUI. (See "Logging In" on page 238)
2. Click on **System > Network > VLAN** on the left navigational pane.
3. Select a subnet and click on the **NAT** tab on the right. All configured NAT rules will be displayed.
4. Click on **+** to activate the **Add NAT Rule** dialogue.
5. Configure outbound NAT using either of the following methods:

   - Enter a **From** IP and the **Up To** IP which specifies the IP range. Also enter the **Out** (outbound NAT IP) address.

   - Enter a **from** IP, without the **Up To** IP.Also enter the **Out** (outbound NAT IP) address.

   The **From** address is the source IP address (or range of addresses) to which this NAT rule applies. Use a CIDR-format IP address to specify a range. If the source IP address of an outbound packet matches this IP address (or falls within the specified range), then the packet is modified to use the IP address specified by the **Out** parameter as the source IP.

   The **Out** address specifies that if the source IP address of an outbound packet matches the IP address (or IP address range) specified by the **From** parameter, then the packet is modified to use this IP address as the source IP.

6. Click on **Commit** to save your settings. The new NAT Rule will appear in the **Subnet NAT** display on the right.

Configuring outbound NAT using the CLI

1. Log in to eqcli as described in "Starting the CLI" on page 142.

2. NAT can be set up by entering a **`from`** parameter in CIDR format that specifies the IP range.

   The **`from`** address is the source IP address (or range of addresses) to which this NAT rule applies. Use a CIDR-format IP address to specify a range. If the source IP address of an outbound packet matches this IP address (or falls within the specified range), then the packet is modified to use the IP address specified by the **`out`** parameter as the source IP.

   The **`out`** address specifies that if the source IP address of an outbound packet matches the IP address (or IP address range) specified by the **`from`** parameter, then the packet is modified to use this IP address as the source IP.

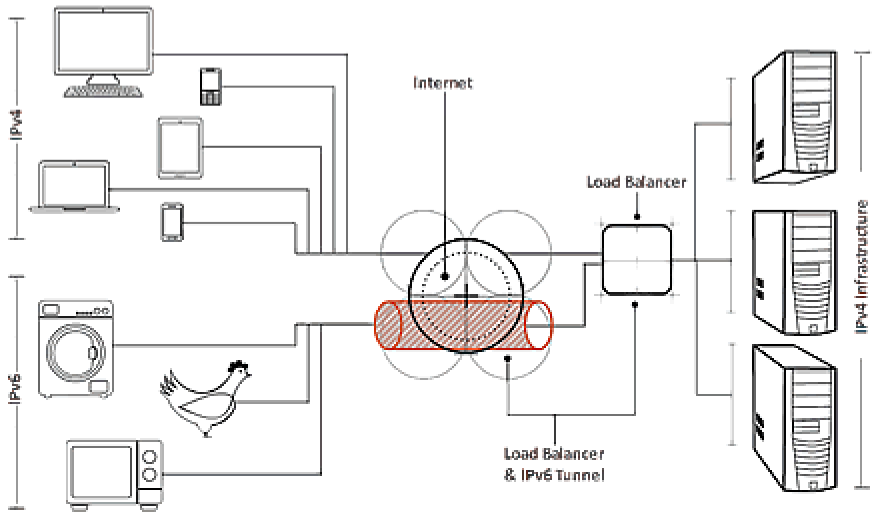
```
eqcli> vlan vlan-name subnet subnet-name nat from ip_cidr out 1.2.3.33 nat
subnet-name out gw gateway ip
```

IPv6 Tunnel Overview

Every network administrator needs to have a strategy to address the transition to the IPv6 Internet. Various *transition mechanisms* have been defined that are intended to make it as easy as possible for organizations to get on the IPv6 Internet using their current IPv4 network infrastructure. For many organizations, the easiest and fastest way to get applications up and running on the IPv6 Internet is to use a transition mechanism called an *IPv6 tunnel*.

One of the most common issues when an organization begins to support IPv6 is how to allow IPv6 enabled devices to communicate over those portions of the network that are not IPv6 enabled. This can include both portions of a corporate intranet as well as Internet connections managed by an Internet Service Provider (ISP) that does not yet provide IPv6 connectivity.

An *IPv6 tunnel* solves this issue by encapsulating IPv6 packets inside IPv4 packets for transmission over IPv4-only connections.



An IPv6 tunnel is obtained through an IPv6 *tunnel broker*. An IPv4 connection is established between a system at your site (in this case, an Equalizer) and a system at the tunnel broker's site. Clusters on Equalizerare assigned IPv6 addresses within the subnet assigned by the tunnel broker. Clients can then access the IPv6 cluster address through the tunnel.

There are a number of tunnel brokers providing IPv6 tunnels to various geographical regions. In general, you should pick a tunnel broker that maintains tunnel servers that are geographically close to your location for best performance.

This chapter describes how you can set up an IPv6 tunnel. Hurricane Electric (HE), one of the leading IPv6 tunnel brokers, provides an easy way to configure a tunnel that uses the *6in4tunneling protocol*. Note that a 6in4 tunnel from any tunnel broker can be used and requires the same basic commands on Equalizer to establish your tunnel -- only the required setup on the tunnel broker's website will be different. Hurricane Electric offers an easy to use web interface that allows you to request and configure a tunnel usually within a few hours.

Note that a number of different tunneling protocols exist, and the tunneling protocols supported vary between tunnel brokers, so check a tunnel broker's website to be sure they support 6in4 tunnels before you request one.

For example, Hurricane Electric provides what they call “regular” tunnels and “BGP” tunnels. For Equalizer, you would choose a “regular” Hurricane Electric tunnel, which is a 6in4 tunnel.

A 6in4 tunnel allows a user to access the IPv6 internet by tunneling over an existing IPv4 connection from an IPv6-enabled host to one of Hurricane Electric's IPv6 routers on the internet. Once a tunnel is established, the IPv6 enabled host sends IPv6 traffic over the local IPv4 network by encapsulating IPv6 packets inside IPv4 packets. These packets are sent to the IPv6 routers operated by the tunnel broker, unencapsulated, and then the IPv6 packets are forwarded to the IPv6 internet.

Note - You can use IPv6 cluster addresses without establishing a tunnel on Equalizer if your organization already has established an IPv6 tunnel and Equalizer can send IPv6 traffic through the local tunnel endpoint. In this configuration, you would simply assign cluster IPv6 addresses from the subnet associated with the already established tunnel and route the IPv6 traffic through the tunnel endpoint. This is done with the standard subnet configuration commands.

Configuring an IPv6 Tunnel

Setting up an IPv6 tunnel on Equalizer is basically a two step process:

1. Configure a VLAN over which Equalizer can reach the IPv4 Internet, and request a "6in4" tunnel from a tunnel broker.
2. After you receive the tunnel configuration information from the broker, set up the tunnel endpoint on Equalizer.

Once the tunnel is configured, you can perform additional tasks required to get Equalizer clusters on the IPv6 Internet, including:

- Assigning cluster IPv6 addresses from the subnet address range provided by the tunnel broker.
- Updating DNS to point to the tunnel broker’s DNS servers.

Creating a "6in4" IPv6 Tunnel using the CLI

1. Configure a VLAN and subnet to use as the local IPv4 endpoint for the tunnel using VLAN context commands (See VLAN and Subnet Commands). Note the following:

- The IPv4 address assigned to the subnet must either be a routable IPv4 address or resolve to a routable IPv4 address via Network Address Translation (NAT) on another device.
- The routable IPv4 address associated with this VLAN is the one that is supplied to the tunnel broker as the local endpoint of the tunnel. Changes to this address must be coordinated with the tunnel broker.
- The ports (both tagged and untagged) that are assigned to this VLAN are the ports on which the IPv6 address block assigned by the tunnel broker will be accessible.

2. Request a "regular" tunnel using Hurricane Electric’s website at:

http://www.tunnelbroker.net

When providing the local IPv4 endpoint address, use the IPv4 address assigned to the VLAN subnet created in Step 1, or its routable NAT address.

Hurricane Electric will set up the tunnel and provide you with the following information:

- The IPv4 and IPv6 addresses for the Hurricane Electric tunnel endpoint.
- The IPv6 address of the default route for the tunnel.
- The IPv6 address block assigned by Hurricane Electric (a /64 prefix subnet).
- The IP addresses of Hurricane Electric's IPv6 and IPv4 DNS servers. (See below)

3. Create the tunnel on Equalizer using the information from the previous step. [For illustration, we use the multi-line command format below; the **tunnel** command can also be entered on a single line.] Enter:

```
eqcli> tunnel HE1
eqcli tl-HE1> type ipip
eqcli tl-HE1> local_endpoint ipv4_addr
eqcli tl-HE1> remote_endpoint ipv4_addr
eqcli tl-HE1> local_address ipv6_addr
eqcli tl-HE1> remote_address ipv6_addr
eqcli tl-HE1> commit
```

The parameter arguments are as follows:

type	Currently, ipip is the only permitted tunnel type.
local_endpoint -	The IPv4 address provided to the tunnel broker as the Equalizer endpoint for the tunnel. It is the IP address that the tunnel broker will use to reach Equalizer. This is either 's VLAN IP on the subnet created in Step 1 or the IP address with which Equalizer's VLAN IP is associated via Network Address Translation (NAT).
remote_endpoint -	The IPv4 address of the tunnel broker side of the tunnel as provided by the tunnel broker.
local_address	The IPv6 address of the Equalizer side of the tunnel; this is assigned by the user and must be within the address range provided by the tunnel broker.
remote_address	The IPv6 address of the tunnel broker side of the tunnel as provided by the tunnel broker.

4. If it does not already exist, configure the VLAN presence on the Equalizer for this tunnel:

```
eqcli> vlan vlan_name vid VLAN_ID flags tunnel
eqcli> vlan vlan_name subnet subnet_name ip local_address default_route ipv6_addr
```

Note the following:

- You can choose any names for the VLAN and subnet.
- The VLAN ID (**vid**) supplied must be appropriate for your network configuration.
- The IPv6 address used for the subnet **ip** parameter must be the same as the **local_address** specified for the **tunnel** command in the previous step.
- The **default_route** parameter must be set to the IPv6 address provided by the tunnel broker as the default tunnel route.
- The VLAN parameters **untagged_ports** and **tagged_ports** are not required when the **tunnel** flag is specified on the **vlan** command line. If they are specified, they will be ignored. [The front panel ports on which the tunnel is accessible are the ports defined for the VLAN that we set up in Step 1.]

You should now be able to assign IPv6 addresses from the tunnel's IPv6 address block as cluster IP addresses on Equalizer, and clients from the IPv6 Internet should be able to connect to them using the cluster's IPv6 address.

Configuring DNS for IPv6 Tunnels

The Domain Name System (DNS) is used by both IPv4 and IPv6 systems to provide name to address mapping, and address to name mapping. Systems that support both IPv4 and IPv6 will require DNS entries that describe the mappings for each protocol.

A new DNS record type, AAAA (sometime referred to as "quad-A"), has been defined for IPv6 name to address mappings (see RFC 3596). AAAA records contain a single IPv6 address mapped to a fully qualified domain name (FQDN). The assigned value for this record type is 28 (decimal). You will need to create AAAA records on your authoritative DNS server in order to access IPv6-enabled systems using their FQDN.

In addition, a special domain rooted at ".IP6.ARPA" is defined in RFC 3596 to provide a way of mapping an IPv6 address to a host name. This is commonly called "DNS reverse lookup", and is specified in your authoritative DNS server's reverse lookup files using PTR (or "pointer") records.

Please refer to the DNS documentation appropriate for your network configuration for more information on adding IPv6 AAAA and PTR records.

Failover

Refer to "Understanding Failover" on page 684 for complete descriptions of failover configurations and setups.

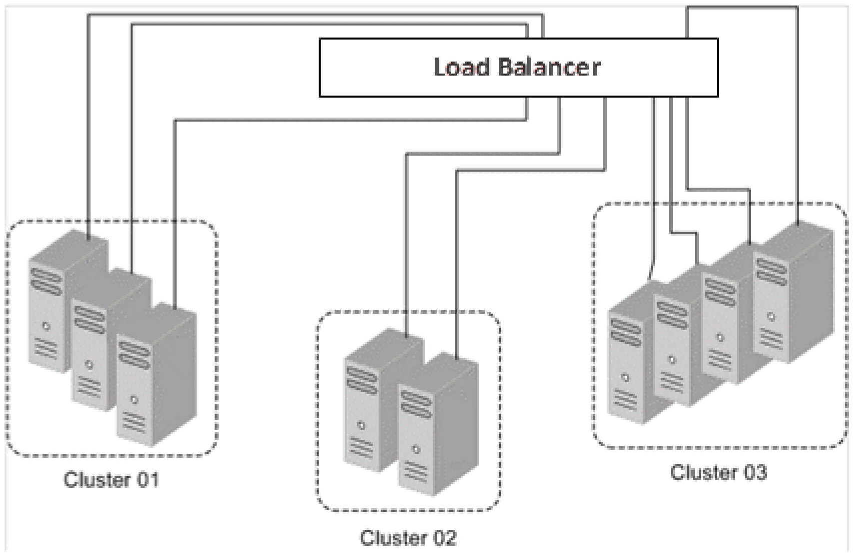
Chapter 12

Working with Clusters and Match Rules

Sections in this chapter include:

Overview

A cluster is a collection of server pools with a single network-visible IP address. All client requests come into Equalizer through a cluster IP address, and are routed by Equalizer to an appropriate server, according to the load balancing options set on the cluster. The figure below shows a conceptual diagram of an Equalizer with three clusters.



Each cluster must be assigned at least one server pool, a grouping of real servers that will be used to respond to incoming client requests. Servers in a server pool are called 'server instances', to distinguish them from the real server definitions with which they are associated (refer to "Managing Server Pools" on page 476 for more information).

The parameters you specify when setting up a virtual cluster determine how client and server connections are managed, and how requests are load balanced among the server instances in a server pool.

Before beginning to define a cluster, you need to do the following:

1. Determine the IP addresses to use for each cluster, and the IP addresses to use for all of your real servers.
2. Determine the cluster types appropriate for your configuration.

Notes:

- The Layer 4 TCP and UDP clusters can use only IPv4 cluster addresses and can only be used with servers that have IPv4 addresses.
- The Layer 7 TCP cluster is used to provide IPv6 addressing for Layer 4 protocols, and can support IPv4 and IPv6 addressing for clusters and servers. The functionality is very much like Layer 4 TCP clusters. This type of cluster should be used when IPv6 addressing is required for a TCP protocol other than HTTP or HTTPS.
- L4 UDP clusters are appropriate for connectionless (stateless) applications, such as DNS, TFTP, Voice over IP (VoIP), and streaming applications -- any application that exchanges short packets with many clients, and where dropped packets are preferred to delayed packets (i.e., the highest possible network performance is required). Layer 4 UDP clusters do not currently support IPv6 addressing.
- Layer 7 HTTPS clusters also provide SSL Offloading: all SSL certificate operations are performed by the cluster, not by the servers behind the cluster, thus improving overall cluster performance.

After you decide on the cluster types you need, you'll then need to determine the additional settings and flags to be used on the cluster and its server pools. For most configuration, it is often a good idea to start with the defaults and make incremental changes as you examine traffic passing through your clusters.

Match Rules

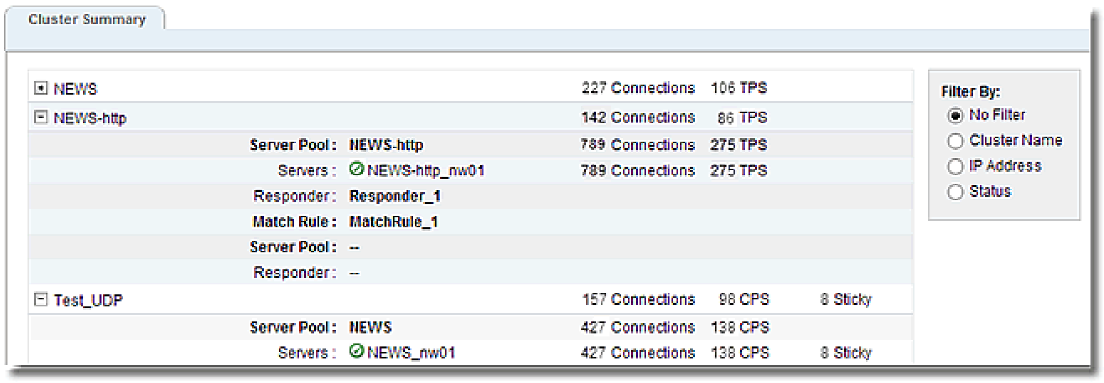
Layer 7 HTTP and HTTPS clusters can use logical constructs called “Match Rules” to control the processing of the incoming data stream from clients. Match rules extend the Layer 7 load balancing capabilities of HTTP and HTTPS clusters by allowing you to define a set of logical conditions which, when met by the contents of the request, trigger the load balancing behavior specified in the match rule.

Cluster Summary

A summary of cluster connection statistics can be displayed using either the GUI or CLI:

Cluster Summary using the GUI

The example of a Cluster Summary screen shown below displays an expandable, sortable summary listing of all of the clusters configured on your Equalizer.This table displays basic status and statistics for the currently configured clusters, their associated server pools, and Layer 7 match rules. Click on the **Load Balance** configuration tab on the left navigational pane if it is not already selected. Then click on **Clusters** to display this summary screen.



Cluster Summary

⊞ NEWS		227 Connections	106 TPS	
⊟ NEWS-http		142 Connections	86 TPS	
Server Pool :	**NEWS-http**	789 Connections	275 TPS	
Servers :	✓ NEWS-http_nw01	789 Connections	275 TPS	
Responder :	**Responder_1**			
Match Rule :	**MatchRule_1**			
Server Pool :	--			
Responder :	--			
⊟ Test_UDP		157 Connections	98 CPS	8 Sticky
Server Pool :	**NEWS**	427 Connections	138 CPS	
Servers :	✓ NEWS_nw01	427 Connections	138 CPS	8 Sticky

Filter By:
[✓ No Filter]
Cluster Name
IP Address
Status

Status Indicators

These icons provide status of the server instances in the attached server pools.

✓ This icon indicates that the server instances in the attached server pool are up and running.

❗ This icon indicates that one or more of the server instances in the attached server pool are down.

❓ This icon indicates that a server instance has been probed DOWN.

Numerical Statistics Displayed

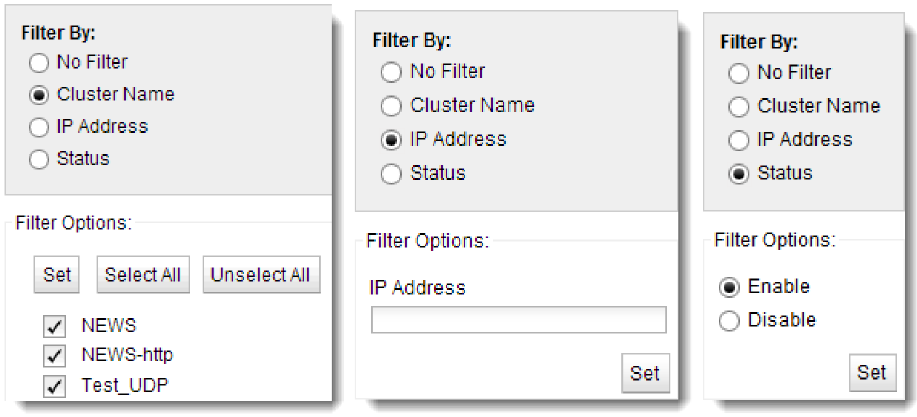
Connections - The number of active (current) connections to this cluster.

TPS - The number of Transactions Per Second being processed by the cluster or match rule.

Sticky - For Layer 4 clusters only. This is the number of entries in the "sticky table" for each server.

Customizing the Display

The cluster summary has 3 display options as shown below:



No Filter - selecting this option will display a cluster summary for all of the clusters configured on your Equalizer.

Filter by Cluster Name - selecting this option will display the cluster summary based on the cluster names that you select with the check boxes. Use the **Select All** or **Unselect All** buttons as necessary.

Filter by IP Address - selecting this option will display the entered IP address of the cluster(s) that you would like displayed. For example if you would like to display cluster summary for IPv4 clusters with IP addresses beginning with "172" you would enter "172.*.*.*" - using a wildcard character (*). For IPv6 clusters you would enter prefix specification such as "2001:218:420::/64". After clicking on the **Set** button, the details for those clusters alone will be displayed.

Filter by Status - selecting this option will display will activate the **Enable/Disable** options. Selecting **Enable** will display only those clusters that do not have problem icons associated with them. Selecting **Disable** will display only those clusters that have problem icons associated with them.

Cluster Summary using the CLI

The Cluster Summary screen shown below displays a summary listing of all of the clusters configured on your Equalizer.

Enter the following:

```
eqcli > show cluster

Name            IP Address      Port    Proto

testUDP         4.6.8.9         80      udp
Test-http       1.3.5.7         80      http
Test_https      2.4.6.8         80      https

eqcli >
```

To display the cluster summary for specific clusters enter The http cluster summary display. It is different than the GUI display in that it reflects only information such as the cluster settings, timeouts, responders and persistence.

```
eqcli > show cluster Testhttps

L7 Cluster Name                 : Test_https
Protocol                        : https
IP Address                      : 2.4.6.8
Port                            : 80
Preferred Peer                  :
VID                             : 1
Client Timeout                  : 10
Server Timeout                  : 60
Connection Timeout              : 10
Sticky Timeout                  : 0
Sticky Netmask                  : 32
Custom Header                   :
CRL                             :
CA Certificate                  :
Cipher Spec                     : AES128-SHA:DES-CBC3-SHA:RC4-SHA:
                                 RC4-MD5:AES256-SHA:!SSLv2
Validation Depth                : 9
Flags                           : allow_utf8, allow_sslv3,
                                 ignore_critical_extns,
                                 allow_t-- press space for mls10,
                                 rewrite_redirects, insert_client_ip
Default Certificate             :
SNI Certificate Objects         :
Server Pool                     :
Responder                       :
Cookie Path                     :
Cookie Domain                   :
```

```
Cookie Age                      : 0
Cookie Generation               : 0
Persist Type                    : coyote_cookie_2

eqcli >
```

Cluster Connection Timeouts

Layer 7 clusters (HTTP / HTTPS) and Layer 4 clusters (TCP / UDP) each use a different set of timeout parameters as described below.

Note - Setting cluster timeouts to arbitrarily high values can have an adverse effect on cluster performance, and can result in the cluster no longer processing traffic. We recommend that you start with default timeout values and adjust the timeouts one by one, in small increments, until you get the timeout behavior that you desire.

HTTP and HTTPS Connection Timeouts

Connections to HTTP and HTTPS clusters are managed closely by Equalizer from the client request to the response from the server. Equalizer needs to manage two connections for every Layer 7 connection request: the client connection from which the request originates, and the connection to the server that is the final destination of the request (as determined by the load balancing policy).

1. Equalizer has an idle timer for the established client connection, a connect timer to establish a server connection, and an idle timer for the established server connection. Only one timeout is in use at any given time. This is a summary of how timeouts are used when a client connects to Equalizer:
2. When a client successfully connects to a Virtual Cluster IP, the client timeout applies from the time the connection is established until the client request headers are completely transmitted. Equalizer parses the client's request, and verifies that the request is a valid HTTP request and that the information needed for load balancing is obtained. In general, this happens at the time that the client headers are completed -- which is indicated by the client sending two blank lines for HTTP 1.0 or 1.1; one blank line for HTTP 0.9. Once the headers are completely transmitted to Equalizer, the client timeout is no longer used.
3. As soon as the Equalizer is done examining the header data, it makes a connection to a server, as determined by the load balancing policy, persistence, or a match rule hit. The amount of time that the Equalizer tries to establish a connection to the server is the connect timeout. Once the server connection is established, the connect timeout is no longer used.
4. After Equalizer establishes a connection with a server, the server timeout is the amount of time Equalizer waits for the next bit of data from the server. Any response from the server restarts the server timeout.

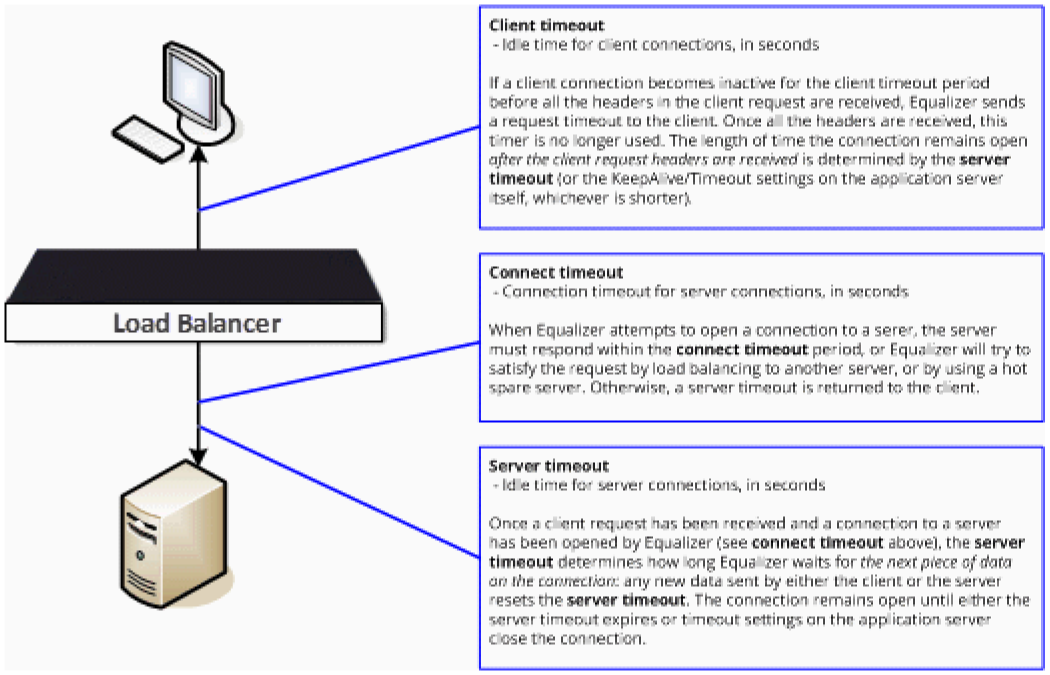
The important distinction between the client timeout and the server timeout is that the client timeout is a “hard” timeout -- the client has the number of seconds specified to transmit all of its headers to Equalizer before Equalizer times out. This is done mainly for security considerations to prevent malicious clients from creating a large number of partial connections and leaking data slowly over the connection, possibly causing resource exhaustion or other undesirable effects on Equalizer.

The **server timeout** by contrast is a “soft” timeout -- the server has the number of seconds specified to send the next piece of information (e.g., the next packet in the sequence). Whenever the client or the server sends a piece of data on the connection, the server timeout is reset. This allows the server to send large data streams in small pieces without timing out, and then close the connection once all the data is sent.

For example, when a client sends a POST operation in a request, the client timeout is used up until the time that the POST headers have all been received. The connect timeout is used until a connection with the server is established. Then, once the connection is established, the server timeout is used for the POST data itself and the subsequent response from the server.

Note that there is the chance that a client will connect, send its headers, and then send continuous data to Equalizer that repeatedly resets the server timeout. This vulnerability is usually avoided by setting a hard client timeout on the application server itself (see "Cluster Connection Timeouts" on page 330).

The figure below summarizes the connection timeout parameters Equalizer uses for Layer 7 client and server connections.



Client timeout
 - Idle time for client connections, in seconds

If a client connection becomes inactive for the client timeout period before all the headers in the client request are received, Equalizer sends a request timeout to the client. Once all the headers are received, this timer is no longer used. The length of time the connection remains open *after the client request headers are received* is determined by the **server timeout** (or the KeepAlive/Timeout settings on the application server itself, whichever is shorter).

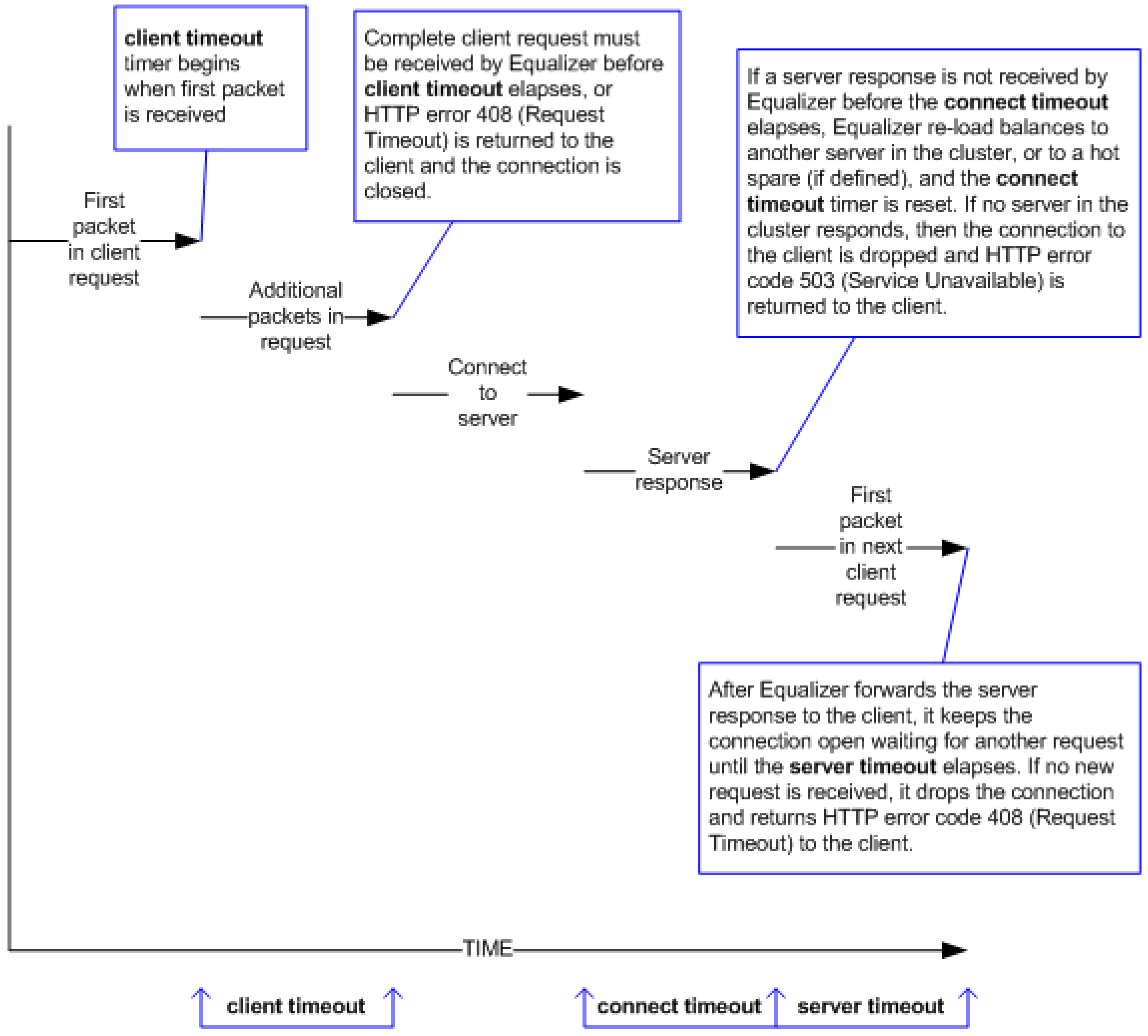
Connect timeout
 - Connection timeout for server connections, in seconds

When Equalizer attempts to open a connection to a serer, the server must respond within the **connect timeout** period, or Equalizer will try to satisfy the request by load balancing to another server, or by using a hot spare server. Otherwise, a server timeout is returned to the client.

Server timeout
 - Idle time for server connections, in seconds

Once a client request has been received and a connection to a server has been opened by Equalizer (see **connect timeout** above), the **server timeout** determines how long Equalizer waits for *the next piece of data on the connection:* any new data sent by either the client or the server resets the **server timeout**. The connection remains open until either the server timeout expires or timeout settings on the application server close the connection.

The timeline below shows the sequence of timeout events when a new connection is received by Equalizer.


client timeout timer begins when first packet is received
Complete client request must be received by Equalizer before **client timeout** elapses, or HTTP error 408 (Request Timeout) is returned to the client and the connection is closed.
If a server response is not received by Equalizer before the **connect timeout** elapses, Equalizer re-load balances to another server in the cluster, or to a hot spare (if defined), and the **connect timeout** timer is reset. If no server in the cluster responds, then the connection to the client is dropped and HTTP error code 503 (Service Unavailable) is returned to the client.
First packet in client request
Additional packets in request
Connect to server
Server response
First packet in next client request
After Equalizer forwards the server response to the client, it keeps the connection open waiting for another request until the **server timeout** elapses. If no new request is received, it drops the connection and returns HTTP error code 408 (Request Timeout) to the client.
TIME
client timeout
connect timeout
server timeout

The following table shows the value range for the Layer 7 HTTP / HTTPS connection timeouts.

Parameter	Minimum	Default	Maximum	Units
client timeout	1.0	5.0	65535.0	seconds
server timeout	1.0	60.0	65535.0	seconds
connect timeout	1.0	10.0	60.0	seconds

The default timeout values are sufficient for many common applications. If timeouts are occurring using the default values, adjust the server timeout to the amount of time you expect your application server to respond to a client request, plus 1 second. If there is high latency between Equalizer and the servers in your cluster, then you may need to increase the connect timeout. The client timeout usually does not need to be changed, but in some situations, HTTPS clusters will require a client timeout between 15 and 30 seconds for best performance. If you do need to increase the client timeout, use the lowest value possible for your configuration to perform well; high values for client timeout increase the risk of denial of service (DoS) attacks.

Once Only Option and HTTP / HTTPS Timeouts

The previous sections describe how the connection timeouts work when the once only flag is disabled on a cluster; that is, when Equalizer is examining every set of headers received on a connection. The once only option, when enabled, specifies that Equalizer will examine only the first set of headers received on a connection. This has the following effects on connection timeouts:

- If you have once only enabled, as soon as the initial transaction (client request and server response) on a connection completes, the connection goes into "streaming" mode and the client timeout is no longer used for this connection. Equalizer does not parse any additional client requests received on the connection. The server timeout is used for the remainder of the connection, and is reset whenever data is received from either side of the connection.
- If you have once only disabled as described in the previous sections, and multiple requests are being sent on the same connection, the client timeout starts counting down again as soon as a new request is received from the client.

Layer 4 Connection Timeouts

Connections to Layer 4 clusters are received by Equalizer and forwarded with little processing. Equalizer simply rewrites the source and/or the destination IP addresses, as appropriate for the cluster, and sends the packet to the server specified by the cluster’s load balancing policy. For Layer 4 TCP clusters, a connection record is kept for each connection so that address translation can be done on the packets going between the servers and clients. The Layer 4 connection timeouts specify how long a connection record is kept by Equalizer.

Layer 4 TCP clusters use the idle timeout and stale timeout parameters that set at cluster levels. The parameters affect how Equalizer manages Layer 4 connection records:

- Connection records need to be removed in cases where the connection is not closed by the client or server, and is left idle. If no data has been received on a connection from either the client or the server after the time period specified by the idle timeout has elapsed, then Equalizer removes the connection record for that connection. Any data received from either client or server resets the idle timer.

  Note that when using Direct Server Return (DSR), the time that a connection record is maintained is determined by adding the idle timeout for the cluster to the sticky time . This additional time is necessary when using DSR, since no server responses are routed through Equalizer (and therefore cannot restart the idle timeout to keep the connection open). For more information on DSR, see "Configuring Direct Server Return" on page 424.
- In other cases, a connection may be initiated but never established, so the connection record goes “stale” and must be removed. If a client fails to complete the TCP connection termination handshake sequence or sends a SYN packet but does not respond to the server’s SYN/ACK, Equalizer marks the connection as incomplete. The stale timeout is the length of time that a connection record for an incomplete connection is maintained.

When Equalizer reclaims a connection, it sends a TCP RST (reset) packet to the server, enabling the server to free any resources associated with the connection. (Equalizer does not send a TCP RST to the client when reclaiming a connection.)

Reducing the stale timeout can be an effective way to counter the effects of SYN flood Denial of Service attacks on server resources. A stale timeout of 10.0 (see table below) would be an appropriate value for a site under SYN flood attack.

Parameter	**Minimum**	**Default**	**Maximum**	**Units**
idle timeout	1.0	60.0	65535.0	seconds
stale timeout	1.0	30.0	120.0	seconds

Note that if you change the stale timeout setting while partially established Layer 4 connections are currently in the queue, those connections will be affected by the new setting.

Application Server Timeouts

Keep in mind that the application server running on the physical servers in your cluster may have its own timeout parameters that will affect the length of time the server keeps connections to Equalizer and the client open. For example, an Apache 2 server has two related timeout directives: **TimeOut and KeepAliveTimeout**:

1. The **TimeOut** directive currently defines the amount of time Apache will wait for three things:
   a. The total amount of time it takes to receive a GET request.
   b. The amount of time between receipt of TCP packets on a POST or PUT request.
   c. The amount of time between ACKs on transmissions of TCP packets in responses.
2. The **KeepAliveTimeout** directive specifies the number of seconds Apache will wait for a subsequent request before closing the connection. Once a request has been received, the timeout value specified by the Timeout directive applies.

In general, if you want Equalizer to control connection timeouts, you must make sure that any timeouts set on the application server are of longer duration than the values set on Equalizer.

For example, with respect to the Apache server timeouts above, the client timeout (for Layer 7 connections) or the idle timeout (for Layer 4 connections) should be of shorter duration than the timeouts set for Apache.

Similarly, the Layer 7 server timeout and connect timeout on Equalizer should be of shorter duration than the TCP connection timeouts set on the servers.

Connection Timeout Kernel Variables

Equalizer uses a number of kernel variables to track connection timeouts, as shown in the table below. You can use the sysctl command to display kernel variables. The two basic formats of this command are:

`sysctl variable_name` Displays the kernel variable variable_name.

`sysctl -a > file` Displays all kernel statistics. This is a long list, so we recommend capturing the list to a file.

eq.idle_timeout	The current setting of the Layer 4 global networking idle timeout parameter.
eq.idle_timedout_count	A Layer 4 counter incremented when a connection is terminated because the idle timeout expired.
eq.stale_timeout	The current setting of the Layer 4 global networking stale timeout parameter.
eq.l7lb.timeouts	The total number of Layer 7 connections dropped because a connection timer expired.
eq.l7lb.http.client_timeouts	The total number of Layer 7 (HTTP and HTTPS) connections that were terminated because the client timeout expired.
eq.l7lb.http.connect_timeouts	The total number of Layer 7 (HTTP and HTTPS) connections that were terminated because the connect timeout expired.
eq.l7lb.http.server_timeouts	The total number of Layer 7 (HTTP and HTTPS) connections that were terminated because the server timeout expired.

Note that there are also some kernel variables associated with Secure Socket Layer (ssl) client connections, such as when someone logs into Equalizer over an SSH connection. These variables are not incremented by HTTPS connections:

```
eq.l7lb.ssl.total_clients
eq.l7lb.ssl.current_clients
eq.l7lb.ssl.max_clients
eq.l7lb.ssl.requests
```

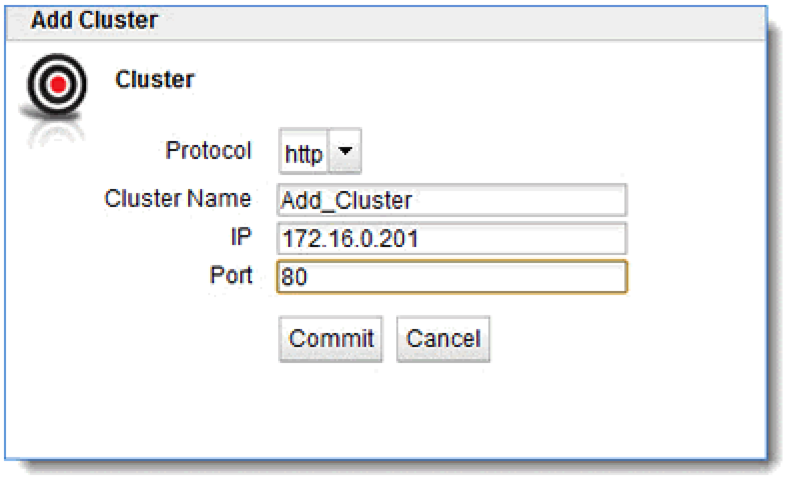
Adding and Deleting Clusters

You can add and delete clusters using either the CLI or the GUI.

Using the GUI

Follow these steps to add a new Layer 7 or Layer 4 virtual cluster using the GUI:

1. Log into the GUI using a log in that has add/del access for global parameters (See "Logging In" on page 238)
2. Right click on Equalizer at the top of the left frame, and select Add Cluster from the menu that appears. The **Add Cluster** form appears as shown below.



Protocol	http
Cluster Name	Add_Cluster
IP	172.16.0.201
Port	80

3. Select **http**, **https**, **tcp**, **L7tcp**or **udp** from the **Protocol** drop down list.
4. Enter the following on the form:

Cluster Name - The logical name for the cluster, or accept Equalizer's default. Each cluster must have a unique name that begins with an alphabetical character. The cluster name is limited to 63 characters.

IP - Enter the cluster IP address, which is the dotted decimal IP address of the cluster. The IP address of the cluster is the external address (for example, 172.16.0.201) with which clients connect to the cluster.

Port - Enter the cluster port: the numeric port number on the Equalizer to be used for traffic between the clients and the cluster. For HTTP clusters, the cluster port defaults to 80. For HTTPS clusters, the cluster port defaults to 443. For TCP ports the cluster port defaults to 88. For UDP ports the cluster port defaults to 53.

5. Click on **Commit** to save the cluster. The new cluster will appear on the **Cluster** branch of the left navigation pane of the GUI.

6. Clicking on each cluster on the Cluster branch of the navigation pane will display the **Configuration Summary**.

Deleting clusters using the GUI

1. Log into the GUI using a login that has add/del access for global parameters (See "Logging In" on page 238)
2. Do one of the following:
   a. Right click on a cluster on the left navigational pane and select **Delete Cluster**.
   b. Click on a cluster on the left navigational pane and drag to the **Delete** (Trash) icon.

Using the CLI

Adding a Cluster

Add a cluster using eqcli as follows. In this example a Layer 7 HTTPS cluster is created. Since the protocol is HTTPS, port 443 is used.

1. Log in to eqcli as described in "Starting the CLI" on page 142.
2. Enter the following at the CLI prompt:

```
eqcli >cluster [clustername] proto protocol ip [xxx.xx.x.xxx] port xxx
```

Refer to "Cluster and Match Rule Commands" on page 171 for additional information on cluster commands in the CLI.

Deleting a Cluster

Do the following to delete a cluster using eqcli as follows:

1. Enter the following at the CLI prompt:

```
eqcli > no cluster [clustername]
```

Modifying a Layer 4 TCP or UDP Cluster

The configuration tabs for a cluster are displayed automatically when a cluster is added to the system, or by selecting the cluster name from the left frame Configuration Tree.

To update the settings on any tab, make changes and select the **Commit** button to save them.

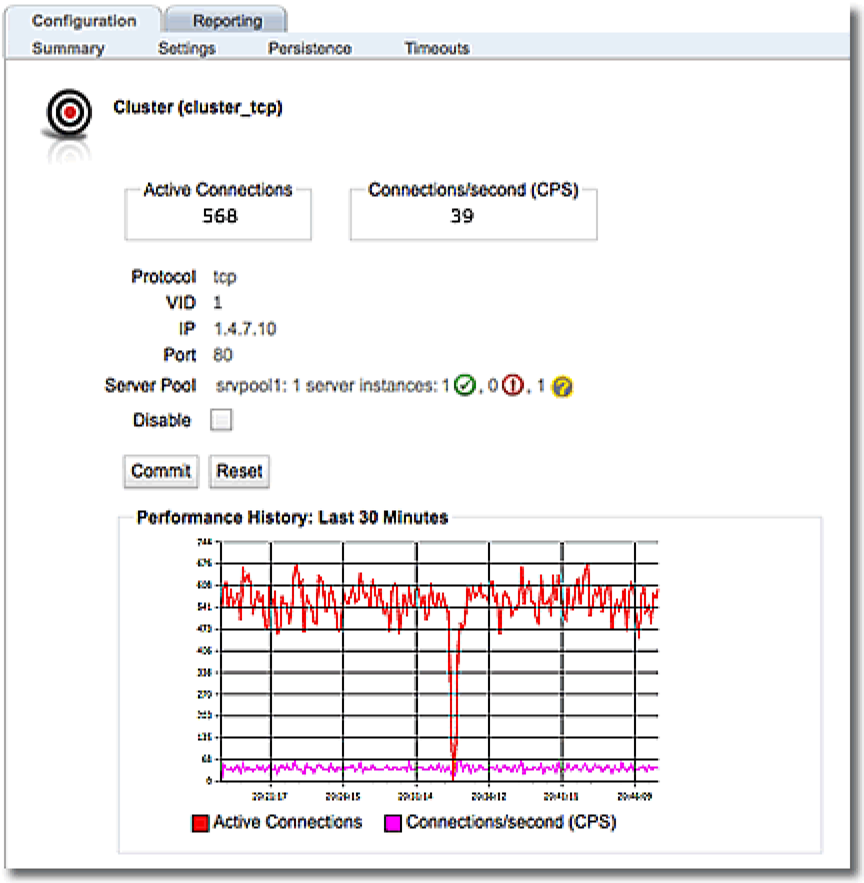
TCP Cluster Configuration Summary

The **TCP Cluster Configuration Summary** screen is displayed automatically when a cluster is added to the system, or by selecting the cluster name from the Cluster branch on the left navigation pane and selecting the **Configuration Summary** tabs. This screen displays a snapshot of the cluster and all of its associated objects (i.e., server pools, server instances and responders), the status of the objects, the Active Connections, Connections/Second and Transactions/Second.

A graphical plot is also displayed showing the traffic flow through the cluster from the past 30 minutes.

In addition you have the option to **Disable** the cluster by selecting the **Disable** check box.

Sample of a TCP Cluster Configuration Summary (GUI)



Cluster (cluster_tcp)

Active Connections 568

Connections/second (CPS) 39

Protocol tcp
VID 1
IP 1.4.7.10
Port 80
Server Pool srvpool1: 1 server instances: 1 ✓, 0 ⓘ, 1 ?
Disable ☐

Sample of a TCP Cluster Configuration Summary (CLI)

The following is an example of a Cluster configuration display using the CLI. Enter `eqcli >` **`show cluster`** ***`clustername`*** **.**

```
eqcli > show cluster cluster_tcp
L4 Cluster Name  : cluster_tcp
Protocol         : tcp
IP Address       : 1.4.7.10
Port             : 80
Port Range       : 0
Preferred Peer   :
VID              : unassigned
Server Pool      : testserverpool
Sticky Timeout   : 0
Sticky Netmask   : 32
Idle Timeout     : 60
Stale Timeout    : 30
Flags            :
eqcli >
```

TCP Cluster Configuration Settings

The following describes the process of entering TCP cluster configuration settings.

Parameters

The table below shows the parameters used with the configuration of a TCP cluster.

GUI Parameter (CLI Parameter)	Description
Protocol **(proto)**	The protocol used for the cluster.
VID **(vid)**	The VLAN ID number. This is an integer between 1 and 4095.
IP **(ip)**	Enter the **IP address**, which is the dotted decimal IP address of the cluster. The IP address of the cluster is the external address (for example, `199.146.85.0`) with which clients connect to the cluster.
Port **(port)**	For TCP protocol clusters the numeric port number on the Equalizer to be used for traffic between the clients and the cluster. This port also becomes the default port for servers added to the cluster (though servers can use a different port number than the one used by the cluster).
Preferred Peer **(preferred_peer)**	Used with N+1 Failover Configuration. (See "Configuring N+1 Failover Between Two EQ/OS 10 Systems" on page 735)
Server Pool **(srvpool)**	The drop down list selects the **Server Pool** (grouping of server instances) to be associated with the TCP cluster.
Range **(range)**	For L4 UDP and L4 TCP protocol clusters, a port **Range** can be defined by entering a value higher than the L4 port configured for the cluster. This range allows Equalizer users to create a single cluster to control the traffic for multiple, contiguous ports.
Flags	
Direct Server Return **(dsr)**	When enabled, Equalizer forwards packets to the server in such a way that the server responds directly to the client, rather than through Equalizer. This option requires special configuration on the cluster; see "Configuring Direct Server Return" on page 424 before enabling this option. The **spoof** option must also be enabled when this option is enabled.

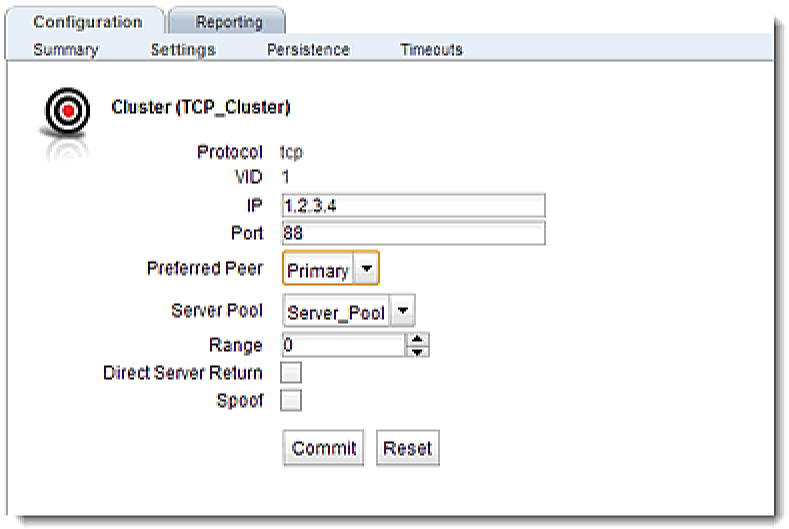
GUI Parameter (CLI Parameter)	Description
Spoof **(spoof)**	When the **Spoof** option is enabled on a cluster, Equalizer uses the client's IP address as the source IP address in all packets sent to a server in that cluster. When **Spoof** is enabled, all server responses to client requests that came through the Equalizer cluster IP address must be routed by the server back to the client through Equalizer. In many cases, the easiest way to do this is to set the default gateway on the server instances in the server pool on a cluster to Equalizer's IP address on the server VLAN. If this is not possible, you can establish static routes on the server to send responses to specific client IP addresses to Equalizer's IP address on the VLAN. If you disable **Spoof**, the server receiving the request will see Equalizer's IP address as the client address because the TCP connection to the client is terminated when the request is routed. The server will therefore send its response back to Equalizer's IP address. When the **Spoof** flag is disabled on a Layer 4 cluster: If there is more than one VLAN defined, all server instances on a server pool *must* be located on the second defined VLAN in the configuration (the VLAN that appears after the **Default** VLAN in the GUI, in **ifconfig** output, and in the configuration file), so that source NAT will work correctly. This is because the source IP address used when **spoof** is disabled is the Equalizer IP address on that VLAN.

GUI Parameter (CLI Parameter)	Description
Protocol **(proto)**	The protocol used for the cluster.
VID **(vid)**	The VLAN ID number. This is an integer between 1 and 4095.
IP **(ip)**	Enter the **IP address**, which is the dotted decimal IP address of the cluster. The IP address of the cluster is the external address (for example, `199.146.85.0`) with which clients connect to the cluster.
Port **(port)**	For TCP protocol clusters the numeric port number on the Equalizer to be used for traffic between the clients and the cluster. This port also becomes the default port for servers added to the cluster (though servers can use a different port number than the one used by the cluster).
Preferred Peer **(preferred_peer)**	Used with N+1 Failover Configuration. (See "Configuring N+1 Failover Between Two EQ/OS 10 Systems" on page 735)
Server Pool **(srvpool)**	The drop down list selects the **Server Pool** (grouping of server instances) to be associated with the TCP cluster.

GUI Parameter (CLI Parameter)	Description
Range **(`range`)**	For L4 UDP and L4 TCP protocol clusters, a port **Range** can be defined by entering a value higher than the L4 port configured for the cluster. This range allows Equalizer users to create a single cluster to control the traffic for multiple, contiguous ports.
Flags	
Direct Server Return **(`dsr`)**	When enabled, Equalizer forwards packets to the server in such a way that the server responds directly to the client, rather than through Equalizer. This option requires special configuration on the cluster; see "Configuring Direct Server Return" on page 424 before enabling this option. The **spoof** option must also be enabled when this option is enabled.
Spoof **(`spoof`)**	When the **Spoof** option is enabled on a cluster, Equalizer uses the client's IP address as the source IP address in all packets sent to a server in that cluster. When **Spoof** is enabled, all server responses to client requests that came through the Equalizer cluster IP address must be routed by the server back to the client through Equalizer. In many cases, the easiest way to do this is to set the default gateway on the server instances in the server pool on a cluster to Equalizer's IP address on the server VLAN. If this is not possible, you can establish static routes on the server to send responses to specific client IP addresses to Equalizer's IP address on the VLAN. If you disable **Spoof**, the server receiving the request will see Equalizer's IP address as the client address because the TCP connection to the client is terminated when the request is routed. The server will therefore send its response back to Equalizer's IP address. When the **Spoof** flag is disabled on a Layer 4 cluster: If there is more than one VLAN defined, all server instances on a server pool *must* be located on the second defined VLAN in the configuration (the VLAN that appears after the **Default** VLAN in the GUI, in **ifconfig** output, and in the configuration file), so that source NAT will work correctly. This is because the source IP address used when **spoof** is disabled is the Equalizer IP address on that VLAN.

TCP Cluster Configuration Settings using the GUI

The TCP Cluster Settings screen for a TCP cluster is displayed by selecting a cluster and from the left navigational pane and then selecting the **Configuration Settings** tabs.



Cluster (TCP_Cluster)

Protocol	tcp
VID	1
IP	1.2.3.4
Port	88
Preferred Peer	Primary
Server Pool	Server_Pool
Range	0
Direct Server Return	☐
Spoof	☐

Click on the **Commit** button after making changes.

TCP Cluster Configuration Settings Using the CLI

TCP Cluster configuration settings can be configured after creating a TCP Cluster. At the CLI prompt you can enter parameters either globally or in cluster context. The following format should be used.

```
eqcli > cluster clustername parameter value flags flag
```

Use the table above to configure the parameters that can be used.

Refer to "Cluster and Match Rule Commands" on page 171 for additional information on using cluster commands in the CLI.

TCP Cluster Persistence

The TCP Cluster Configuration Persistence screen is used to configure **Sticky Netmask** values, Timeouts and assign the Inter Cluster sticky flag to the selected TCP cluster.

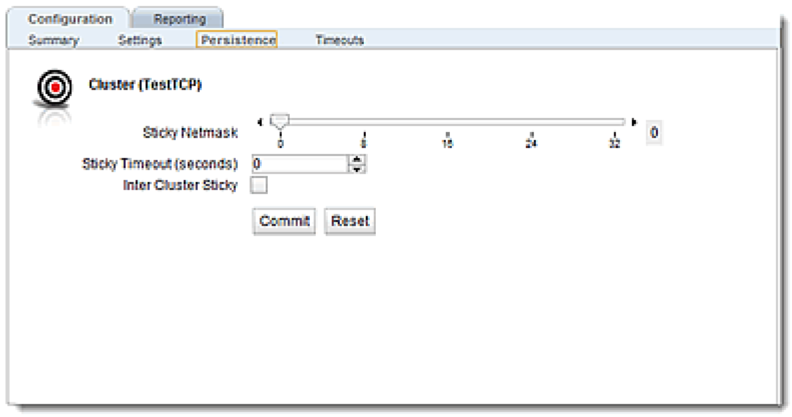
Parameters

The table below shows the persistence parameters used with TCP clusters.

GUI Parameter (CLI Parameter)	Description
Sticky Netmask **(`sticky_netmask`)**	Enables sticky network aggregation for a subnet. Sticky network aggregation is applicable for Layer 4 and Layer 7 clusters. Sticky network aggregation enables Equalizer to correctly handle sticky connections from ISPs that use multiple proxy servers to direct user connections. When you enable sticky network aggregation, all the connections coming from a particular network are directed to the same server. (Typically, all the servers in a proxy farm are on the same network.) The sticky netmask value indicates which portion of the address Equalizer should use to identify particular networks. Values are: **0-32** for IPV4 clusters (default=32) **0-128** for IPV6 clusters
Sticky Time Out **(`stickyto`)**	Sticky Timeout is the number of seconds that Equalizer should "remember" connections from clients. Valid values are from 0 (which disables sticky connections) to 1073741823 seconds (or over 34 years). For more information, refer to "Enabling Sticky Connections" on page 413.
Flags	
Inter-Cluster Sticky **(`ics`)**	With the **inter-cluster sticky** option, you can configure Equalizer to direct requests from a client to the same server on any available port that has a current persistent connection in any cluster.Inter-Cluster Sticky is a Layer 4 option that allows you to extend Layer 4 persistence across multiple server ports.

Configure TCP Cluster Persistence Using the GUI:

TCP Cluster Persistence can be accessed by selecting a cluster from the left navigational pane and selecting the **Configuration > Persistence** tab. The following will be displayed.



Sticky Netmask: 0
Sticky Timeout (seconds): 0
Inter Cluster Sticky

Configured TCP Cluster Persistence parameters using the table above. Click on the **Commit** button after making changes to the settings.

Configure TCP Cluster Persistence Using the CLI

TCP Cluster Persistence can be accessed by selecting configuring a cluster and configuring parameters on the CLI command line either globally or in cluster context. Use the following format:

```
eqcli > cluster cluster clustername parameter value flags flag
```

Where:

clustername - is the the name fo the cluster.

parameter - is the parameter.

value - is the value associated with the parameter.

flag - is the flag to be associated with the cluster.

Use the table above for parameters and values .

Refer to "Cluster and Match Rule Commands" on page 171 for additional information on using cluster commands in the CLI.

TCP Cluster Timeouts

The TCP Cluster Configuration Timeouts screen is used to configure the various timeouts shown below for the selected TCP cluster.

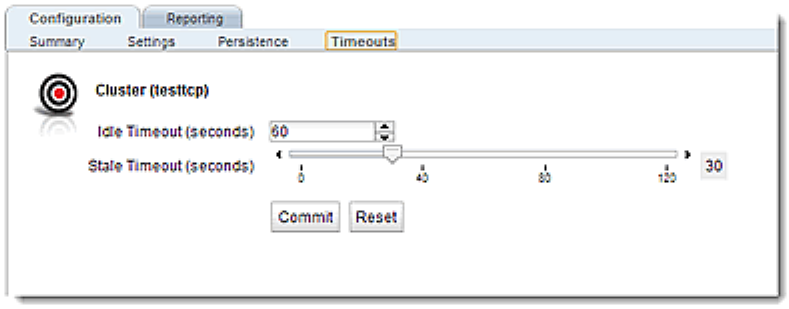
Parameters

The table below shows the cluster timeouts used with the configuration of a TCP cluster.

GUI Parameter (CLI Parameter)	Description
Idle Timeout (`idleto`)	The idle timeout (default:60) can be set at the global and cluster levels, while stale timeout can be set at the global level only. Connection records need to be removed in cases where the connection is not closed by the client or server, and is left idle. If no data has been received on a connection from either the client or the server after the time period specified by the idle timeout has elapsed, then the connection record for that connection is removed. Any data received from either client or server resets the idle timer.(1-65535)
Stale Timeout (`staleto`)	The stale timeout (default:30) is the length of time that a connection record for an incomplete connection is maintained. (1-120)

TCP Cluster Timeouts Using the GUI

The TCP Cluster Timeout screen can be accessed by selecting a cluster from the left navigational pane and selecting the **Configuration** > **Timeouts** tabs.



Idle Timeout (seconds) 60
Stale Timeout (seconds) 30

Use the table above for parameters and values. Click on the **Commit** button after making changes to the settings.

TCP Cluster Timeouts Using the CLI

The TCP Cluster Timeouts can be configured using the CLI either globally or in cluster context. Configure the timeouts using the following format:

```
eqcli > cluster clustername {idleto|staleto} value
```

Use the table above for parameters and values.

Where:

clustername - is the the name fo the cluster.

value - is the value associated with the timeout.

Use the table above for parameters and values .

Refer to "Cluster and Match Rule Commands" on page 171 for additional information on using cluster commands in the CLI.

UDP Cluster Configuration Summary

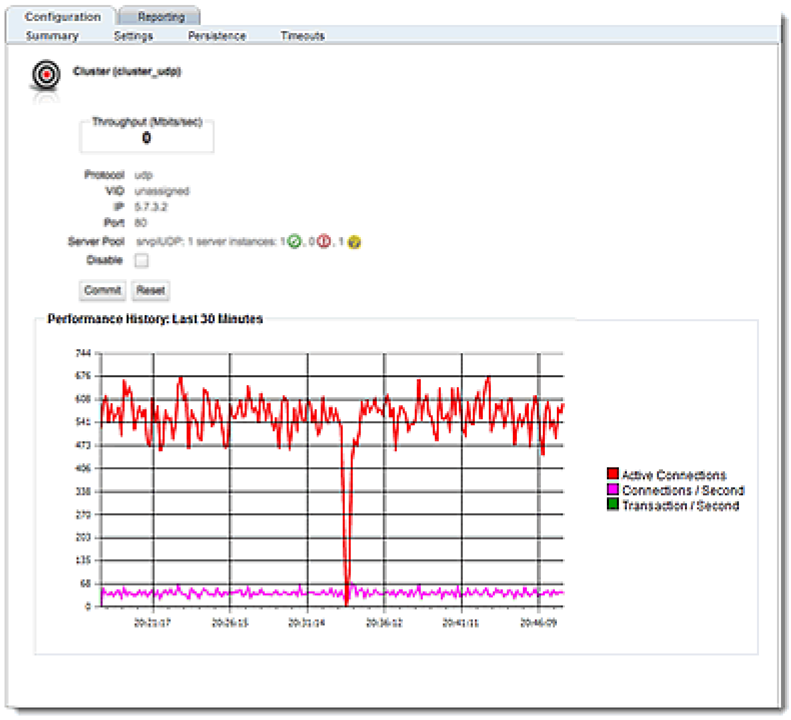
UDP Cluster Configuration Summary can be displayed using either the GUI or the CLI.

Sample of UDP Cluster Configuration Summary on the GUI

The UDP **Cluster Configuration Summary** screen is displayed automatically when a UDP cluster is added to the system, or by selecting the cluster name from the **Cluster** branch on the left navigation pane. This screen displays a snapshot of the cluster and all of its associated objects (i.e., server pools, server instances and responders) and the status of the objects.

A graphical plot is also displayed showing the traffic flow through the cluster from the past 30 minutes.

In addition you have the option to **Disable** the cluster by selecting the **Disable** check box.Note that if a connection is active and the cluster is disabled, then any packets received are dropped. The connection will eventually timeout and be removed.



Cluster (cluster_udp)

Throughput (Mbits/sec)
0

Protocol	udp
VID	unassigned
IP	5.7.3.2
Port	80
Server Pool	srvpUDP: 1 server instances: 1 ✓, 0 ⓘ, 1 ●
Disable	☐

Performance History: Last 30 Minutes

🟥 Active Connections
🟪 Connections / Second
🟩 Transaction / Second

Sample of UDP Cluster Summary Using the CLI

Select a UDP Cluster and enter the following in the CLI.either globally or in cluster context.

```
eqcli > show cluster cluster_udp
L4 Cluster Name  : cluster_udp
Protocol         : udp
IP Address       : 5.7.3.2
Port             : 80
Port Range       : 0
Preferred Peer   :
VID              : unassigned
Server Pool      : srvplUDP
Sticky Timeout   : 0
Sticky Netmask   : 32
Stale Timeout    : 30
Flags            :

eqcli >
```

UDP Cluster Configuration Settings

You can configure UDP Clusters using either the GUI or the CLI.

Parameters

The table below shows the parameters and values used with the configuration of a UDP cluster.

GUI Parameter (CLI Parameter)	Description
Protocol **(proto)**	The protocol used for the cluster.
VID **(vid)**	The VLAN ID number. This is an integer between 1 and 4095.
IP **(ip)**	Enter the **IP address**, which is the dotted decimal IP address of the cluster.
Port **(port)**	For TCP protocol clusters the numeric port number on the Equalizer to be used for traffic between the clients and the cluster. For TCP clusters, the port defaults to 80. This port also becomes the default port for servers added to the cluster (though servers can use a different port number than the one used by the cluster).
Preferred Peer **(preferred_peer)**	Used with N+1 Failover Configuration. Refer to "Configuring N+1 Failover Between Two EQ/OS 10 Systems" on page 735.
Server Pool **(srvpool)**	The drop down list selects the **Server Pool** (grouping of server instances) to be associated with the TCP cluster.
Range **(range)**	For L4 UDP and L4 TCP protocol clusters, a port **Range** can be defined by entering a value higher than the L4 port configured for the cluster. This range allows Equalizer users to create a single cluster to control the traffic for multiple, contiguous ports.
Flags	
Direct Server Return **(dsr)**	When enabled, Equalizer forwards packets to the server in such a way that the server responds directly to the client, rather than through Equalizer. This option requires special configuration on the cluster; see "Configuring Direct Server Return" on page 424 before enabling this option. The **spoof** option must also be enabled when this option is enabled.

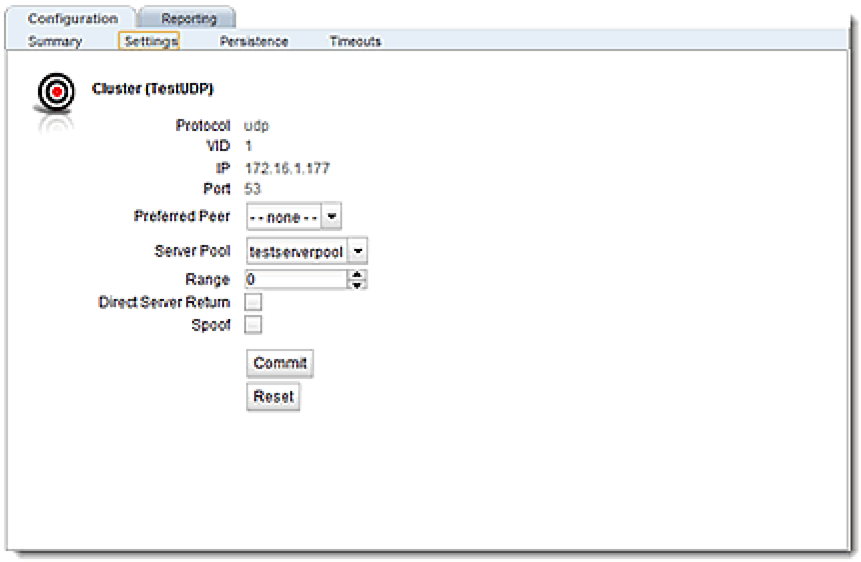
GUI Parameter (CLI Parameter)	Description
Spoof **(spoof)**	When the **Spoof** option is enabled on a cluster, Equalizer uses the client's IP address as the source IP address in all packets sent to a server in that cluster. When **Spoof** is enabled, all server responses to client requests that came through the Equalizer cluster IP address must be routed by the server back to the client through Equalizer. In many cases, the easiest way to do this is to set the default gateway on the server instances in the server pool on a cluster to Equalizer's IP address on the server VLAN. If this is not possible, you can establish static routes on the server to send responses to specific client IP addresses to Equalizer's IP address on the VLAN. If you disable **Spoof**, the server receiving the request will see Equalizer's IP address as the client address because the TCP connection to the client is terminated when the request is routed. The server will therefore send its response back to Equalizer's IP address. When the **Spoof** flag is disabled on a Layer 4 cluster: If there is more than one VLAN defined, all server instances on a server pool *must* be located on the second defined VLAN in the configuration (the VLAN that appears after the **Default** VLAN in the GUI, in **ifconfig** output, and in the configuration file), so that source NAT will work correctly. This is because the source IP address used when **spoof** is disabled is the Equalizer IP address on that VLAN.

GUI Parameter (CLI Parameter)	Description
Protocol **(proto)**	The protocol used for the cluster.
VID **(vid)**	The VLAN ID number. This is an integer between 1 and 4095.
IP **(ip)**	Enter the **IP address**, which is the dotted decimal IP address of the cluster.
Port **(port)**	For TCP protocol clusters the numeric port number on the Equalizer to be used for traffic between the clients and the cluster. For TCP clusters, the port defaults to 80. This port also becomes the default port for servers added to the cluster (though servers can use a different port number than the one used by the cluster).
Preferred Peer **(preferred_peer)**	Used with N+1 Failover Configuration. Refer to "Configuring N+1 Failover Between Two EQ/OS 10 Systems" on page 735.
Server Pool **(srvpool)**	The drop down list selects the **Server Pool** (grouping of server instances) to be associated with the TCP cluster.
Range **(range)**	For L4 UDP and L4 TCP protocol clusters, a port **Range** can be defined by entering a value higher than the L4 port configured for the cluster. This range allows Equalizer users to create a single cluster to control the traffic for multiple, contiguous ports.

GUI Parameter (CLI Parameter)	Description
Flags	
Direct Server Return **(dsr)**	When enabled, Equalizer forwards packets to the server in such a way that the server responds directly to the client, rather than through Equalizer. This option requires special configuration on the cluster; see "Configuring Direct Server Return" on page 424 before enabling this option. The **spoof** option must also be enabled when this option is enabled.
Spoof **(spoof)**	When the **Spoof** option is enabled on a cluster, Equalizer uses the client's IP address as the source IP address in all packets sent to a server in that cluster. When **Spoof** is enabled, all server responses to client requests that came through the Equalizer cluster IP address must be routed by the server back to the client through Equalizer. In many cases, the easiest way to do this is to set the default gateway on the server instances in the server pool on a cluster to Equalizer's IP address on the server VLAN. If this is not possible, you can establish static routes on the server to send responses to specific client IP addresses to Equalizer's IP address on the VLAN. If you disable **Spoof**, the server receiving the request will see Equalizer's IP address as the client address because the TCP connection to the client is terminated when the request is routed. The server will therefore send its response back to Equalizer's IP address. When the **Spoof** flag is disabled on a Layer 4 cluster: If there is more than one VLAN defined, all server instances on a server pool *must* be located on the second defined VLAN in the configuration (the VLAN that appears after the **Default** VLAN in the GUI, in **ifconfig** output, and in the configuration file), so that source NAT will work correctly. This is because the source IP address used when **spoof** is disabled is the Equalizer IP address on that VLAN.

UDP Cluster Configuration Using the GUI

The UDP Cluster **Configuration Settings** screen shown below is displayed automatically when the cluster is added to the system, or by selecting the cluster from the left navigational pane on the GUI and selecting the **Configuration Settings** tabs.



Cluster (TestUDP)

Protocol	udp
VID	1
IP	172.16.1.177
Port	53
Preferred Peer	- - none - -
Server Pool	testserverpool
Range	0
Direct Server Return	☐
Spoof	☐

Use the table above for descriptions of the parameters and values. Click on the **Commit** button after making changes.

UDP Cluster Configuration Using the CLI

UDP Clusters can be configured in the CLI either globally or in cluster context. Enter parameters using the following format.

```
eqcli > cluster clustername parameter value flags flag
```

Use the table above for descriptions of the parameters and values.

Where:

clustername - is the the name fo the cluster.

parameter - is the parameter.

value - is the value associated with the parameter.

flag - is the flag to be associated with the cluster.

Use the table above for parameters and values .

Refer to "Cluster and Match Rule Commands" on page 171 for additional information on using cluster commands in the CLI.

UDP Cluster Persistence

UDP Cluster persistence includes the configuration of Sticky Netmask values, timeouts, and the assignment of an Inter Cluster Sticky flag on the cluster, if necessary.

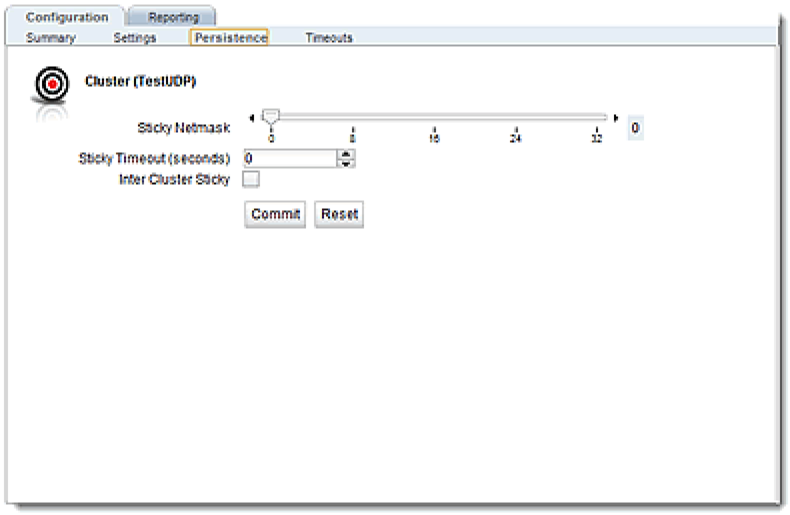
Parameters

The table below shows the Cluster Persistence parameters and values used with the configuration of a UDP cluster.

GUI Parameter (CLI Parameter)	
Sticky Netmask **(sticky_netmask)**	Enables sticky network aggregation for a subnet. Sticky network aggregation is applicable for Layer 4 and Layer 7 clusters. Sticky network aggregation enables Equalizer to correctly handle sticky connections from ISPs that use multiple proxy servers to direct user connections. When you enable sticky network aggregation, all the connections coming from a particular network are directed to the same server. (Typically, all the servers in a proxy farm are on the same network.) The sticky netmask value indicates which portion of the address Equalizer should use to identify particular networks. Values are: **0-32** for IPV4 clusters (default=32) **0-128** for IPV6 clusters
Sticky Time Out **(stickyto)**	Sticky Timeout is the number of seconds that Equalizer should "remember" connections from clients. Valid values are from 0 (which disables sticky connections) to 1073741823 seconds (or over 34 years). For more information, refer to "Enabling Sticky Connections" on page 413.
Flags	
Inter-Cluster Sticky **(ics)**	With the **inter-cluster sticky** option, you can configure Equalizer to direct requests from a client to the same server on any available port that has a current persistent connection in any cluster.Inter-Cluster Sticky is a Layer 4 option that allows you to extend Layer 4 persistence across multiple server ports.

UDP Cluster Persistence Using the GUI

The UDP Cluster **Configuration >Persistence** screen can be accessed by selecting a cluster from the left navigational pane and selecting the **Configuration > Persistence** tab.



Cluster (TestUDP)

Sticky Netmask 0

Sticky Timeout (seconds) 0

Inter Cluster Sticky ☐

Configure using the table above for parameters and values. Click on the **Commit** button after making changes to the settings.

Configuring UDP Cluster Persistence Using the CLI

UDP Cluster persistence can be configured using the CLI in either global format or within the cluster context. Use the following format.

```
eqcli > cluster clustername parameter value flags flag
```

Configure using the table above for parameters and values.

Where:

clustername - is the the name fo the cluster.

parameter - is the parameter.

value - is the value associated with the parameter.

flag - is the flag to be associated with the cluster.

Use the table above for parameters and values .

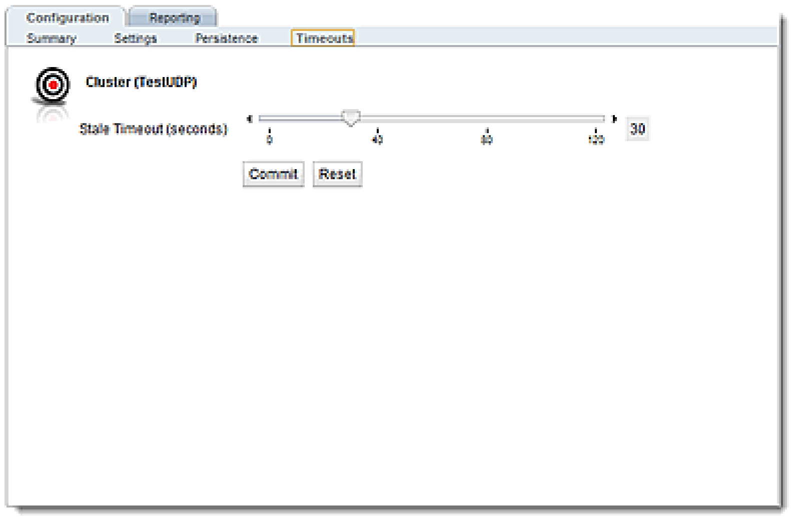
Refer to "Cluster and Match Rule Commands" on page 171 for additional information on using cluster commands in the CLI.

UDP Cluster Timeouts

UDP Cluster Timeouts involve the configuration of a state timeout. A **Stale Timeout** is the length of time in seconds that a partially open or closed Layer 4 connection is maintained. If a client fails to complete the TCP connection termination handshake sequence or sends a SYN packet but does not respond to the server's SYN/ACK, Equalizer marks the connection as incomplete.

Configuring UDP Cluster Timeouts Using the GUI

The UDP Cluster **Configuration Timeouts** screen can be accessed by selecting a cluster from the left navigational pane and selecting the **Configuration Timeouts** tab.



Configure the Stale Timeout and click on the **Commit** button after making changes to the settings.

Configuring UDP Cluster Timeouts Using the CLI

Configure the Stale Timeout for the UDP cluster using the CLI either globally or in the cluster context.

```
eqcli > cluster clustername staleto value
```

Where:

clustername - is the the name fo the cluster.

value - is the value associated with the stale timeout

Use the table above for parameters and values .

Refer to "Cluster and Match Rule Commands" on page 171 for additional information on using cluster commands in the CLI.

UDP Cluster Limitations

Layer 4 UDP clusters are appropriate for connectionless (stateless) applications, such as DNS, TFTP, Voice over IP (VoIP), and streaming applications. UDP applications typically exchange short packets with many clients, and typically provide faster network performance over TCP applications, because UDP applications do not re-transmit dropped packets and do not performing error checking.

Compared to Layer 7 clusters, UDP clusters share the same general limitations of Layer 4 TCP clusters, the most important being:

1. SSL offload is not supported for UDP clusters. If you would like to use a secure UDP application, you must install certificates directly on your physical servers rather than in the UDP cluster.
2. IP-address based persistence is the only persistence type supported.
3. Match Rules are not supported.

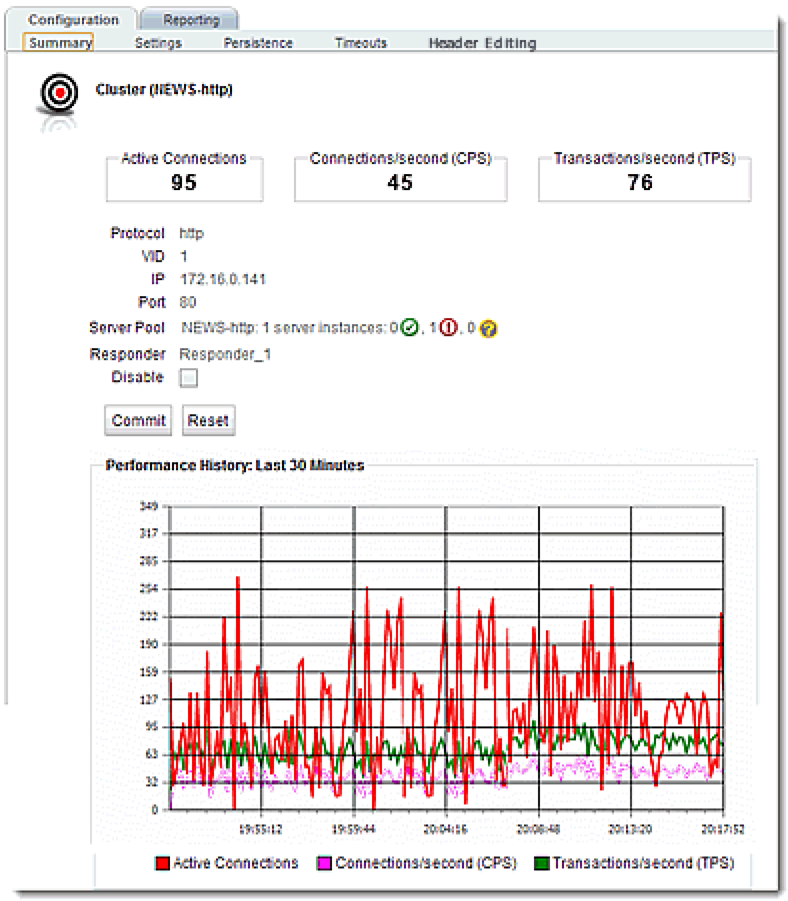
There are also several limitations that apply only to UDP clusters and servers:

1. A UDP server can be used in exactly one UDP cluster. This means that all server pools attached to all UDP clusters must contain UDP servers that each have a unique IP address.
2. UDP clusters can use only IPv4 addresses; all servers used by a UDP cluster must have IPv4 addresses.
3. You can't use ACV probes with UDP servers.

Modifying a Layer 7 HTTP or HTTPS Cluster

On the GUI, the **Configuration Summary** for a layer 7 cluster is displayed automatically when a cluster is added to the system, or by selecting the cluster from **Cluster** branch on the left navigation pane. HTTP and HTTPS clusters parameters are modified using the following tabs:

- **Configuration** including:**Summary**, **Settings**, **Persistence** and **Timeouts**
- **Reporting** including:**Statistics** and **Plotting**



Cluster (NEWS-http)

Active Connections 95
Connections/second (CPS) 45
Transactions/second (TPS) 76

Protocol http
VID 1
IP 172.16.0.141
Port 80
Server Pool NEWS-http: 1 server instances: 0 ✓, 1 ⓘ, 0 ?
Responder Responder_1
Disable ☐

Layer 7 Cluster Configuration Summary

Layer 7 Cluster Summary is a snapshot of the cluster and all of it's associated objects and the status of the objects, as well as configured settings. It can be displayed using either the GUI or the CLI.

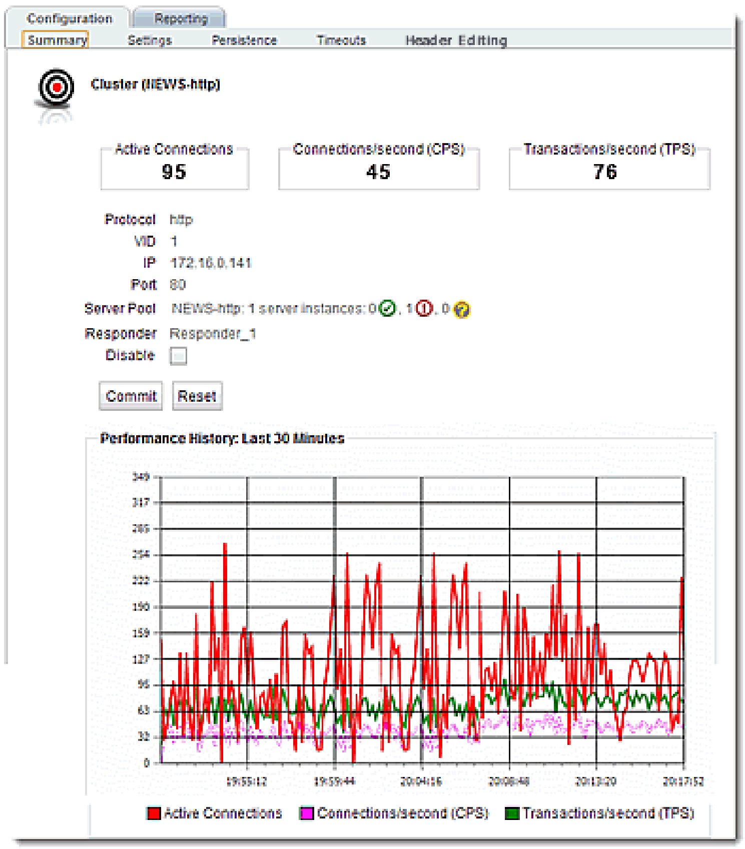
Layer 7 HTTP, HTTPS, and TCP Cluster Configuration Summary Using the GUI

The Layer 7 **Cluster Configuration Summary** screen is displayed automatically as described in "Modifying a Layer 7 HTTP or HTTPS Cluster" on page 362; when a cluster is added to the system, or by selecting the cluster from the **Cluster** branch on the left navigation pane. This screen displays a snapshot of the cluster and all of its associated objects (i.e., server pools, server instances and responders), the status of the objects, the **Active Connections**, **Connections/Second** and **Transactions/Second**.

A graphical plot is also displayed showing the traffic flow through the cluster from the past 30 minutes.

In addition you have the option to **Disable** the cluster by removing its IP address alias from the interface in addition to disabling cluster traffic.

The sample below shows a sample of the **Configuration Summary Screen** for an HTTP cluster. The Summary Screens for the HTTPS and Layer 7 TCP clusters are similar.



Cluster (NEWS-http)

Active Connections	Connections/second (CPS)	Transactions/second (TPS)
95	45	76

Protocol	http
VID	1
IP	172.16.0.141
Port	80
Server Pool	NEWS-http: 1 server instances: 0 ✓, 1 ⓘ, 0 ?
Responder	Responder_1
Disable	☐

Layer 7 HTTP, HTTPS, and TCP Cluster Configuration Summary Using the CLI

Layer 7 HTTP, HTTPS, and TCP Cluster summary can be displayed using the CLI for either globally or in the cluster context. The following is an example:

```
eqcli > show cluster cluster_http
L7 Cluster Name         : cluster_http
Protocol                : http
IP Address              : 4.8.12.16
Port                    : 80
Preferred Peer          :
VID                     : unassigned
Client Timeout          : 10
Server Timeout          : 60
Connection Timeout      : 10
Sticky Timeout          : 0
Sticky Netmask          : 32
Custom Header           :
Flags                   : allow_utf8
Server Pool             : srvpool1
Responder               :
Cookie Path             :
Cookie Domain           :
Cookie Age              : 0
Cookie Generation       : 0
Persist Type            : coyote_cookie_2
Minimum Compression     : 1024
Compression mime_type   : text/
*:application/msword:application/postscript:application/rtf:application/x-
csh:application/x-javascript:application/x-sh:application/x-
shar:application/x-tar:application/x-
tcl:application/xslt+xml:audio/midi:audio/32kadpcm:audio/x-
wav:image/bmp:image/tiff:image/x-rgb
Config Header Edit Script   :
eqcli >
```

Layer 7 HTTP and HTTPS Cluster Settings

The following are descriptions of the functionality and configuration parameters used with Layer 7 HTTP and HTTPS Clusters.

Parameters

The table below shows the parameters, values, and flags used in the configuration of HTTP and HTTPS clusters.

GUI Parameter (CLI Parameter)	Description
Protocol **(`proto`)**	The protocol selected in the Add Cluster form will be displayed "grayed out".
VID **(`vid`)**	The VLAN ID number assigned to the VLAN on which the cluster resides. Refer to Common Networking Scenarios for details.
IP **(`ip`)**	Enter the IP address, which is the dotted decimal IP address of the cluster. The IP address of the cluster is the external address (for example, 172.16.0.201) with which clients connect to the cluster.
Port **(`port`)**	For HTTP and HTTPS protocol clusters, enter the port: the numeric port number on the Equalizer to be used for traffic between the clients and the cluster. For HTTP clusters, the port is normally 80. For HTTPS clusters, the port is normally 443. This port also becomes the default port for servers added to the cluster (though servers can use a different port number than the one used by the cluster).
Preferred Peer **(`preferred_peer`)**	Used with Active-Active Failover. Refer to Configuring Active/Active Failover Between Two Systems.
Server Pool **(`srvpool`)**	A drop down list used to select a Server Pool, or grouping of servers, to which the cluster will communicate with.
Responder **(`resp`)**	A Responder is a server-like object that can be associated with a Match Rule. If an incoming request satisfies a Match Rule expression and all of the servers specified in the Match Rule are down, a Responder definition in the Match Rule (if present) tells Equalizerto send one of two automatic responses to the client.
Custom Header **(`custhdr`)**	A custom HTTP header that Equalizerinserts into all client requests before they are sent to the server. The format of the string is text:text. Also see Specifying a Custom Header for HTTP/HTTPS Clusters.
Compression Minimum Size (ADCs with Hardware Acceleration) **(`compress_min`)**	The minimum file size in bytes required for GZIP compression, if enabled. Equalizeruses GZIP to compress the payload (or content) of the server response before sending it back to the client. This is typically done for 2 reasons: faster client response and better user experience. In addition. less ISP bandwidth is used in sending smaller files back to clients. Files smaller than the minimum specified are not compressed. Default:1024 bytes.

GUI Parameter (CLI Parameter)	Description
Compression MIME Types (ADCs with Hardware Acceleration) (`compress_types`)	Specifies the mime-types that will be compressed when the Compress option is enabled for the cluster. The value of this parameter is a string (maximum length: 512 characters) with valid mime-type names separated by a colon (:). The default compress mime-types string specifies the following mime-types text/* application/msword application/postscript application/rtf application/x-csh application/x-javascript application/x-sh application/x-shar application/x-tar application/x-tcl application/xslt+xml audio/midi audio/32kadpcm audio/x-wav image/bmp image/tiff image/x-rgb
Config Header Edit Script (`hdredit`)	When you create header edits, you place the search and edit functions within one of two meta-functions (**client_request_edit** or **server_ response_edit**) the script will be displayed here.
Flags	
Abort server (`abort_server`)	By default, when a client closes a connection, Equalizerwaits for a response from the server before closing the server connection. If this flag is enabled, Equalizer will not wait for a response before closing the connection to the server; instead it sends a TCP RST (reset) to the server when the client closes the connection. This option will typically reduce the number of server connections in the TIME_WAIT state, as shown by the netstat console command.
Insert client IP (`insert_client_ip`)	When this flag is enabled, Equalizer inserts an X-forwarded-for: header with the client's IP address into all client requests before they are sent to the server. This flag is disabled by default for HTTP clusters and enabled by default for HTTPS clusters.
TCP Multiplexing (`tcp_mux`)	This selection enables TCP multiplexing for a cluster. TCP multiplexing must also be enabled on at least one server instance in the server pool assigned to the cluster (or one of its match rules).
Allow Multibyte Characters (`allow_utf8`)	By default, support for extended characters (8-bit ASCII and multibyte UTF characters) in URIs is disabled. Equalizer returns a 400 Bad Request error when a request URI contains 8-bit or multibyte characters. To enable support for 8-bit and multibyte characters in URIs, click this checkbox. There are potential risks to enabling this option, because it allows Equalizer to pass requests that violate RFC2396; load-balanced servers may be running software that is incapable of handling such requests. Therefore, ensure that your server software is capable of handling URIs containing extended characters and will not serve as a potential weak point in your network before you enable extended characters.

GUI Parameter (CLI Parameter)	Description
Compress (Not applicable to E250GX) **(compress)**	When this option is enabled, Equalizer automatically detects requests to the cluster from compression-capable browser clients and performs GZIP compression on all cluster responses sent to that client. The cluster IP address will not accept requests when this flag is enabled. When enabled, hardware compression will be used on ADCs with compression hardware on board and software compression will be used on ADCs without compression hardware. To view the compression type used on your appliance, do one of the following: 1. At the CLI prompt, enter; **eqcli > version** . The compression type used is displayed beside **Features.** 2. In the GUI, click on the **System** configuration tab, expand the **Global** branch and click on **Dashboard**. The compression type will be displayed beside **Features** in the **System Information** widget on the right. NOTE: If downgrading to a version less than 10.3.x, and the compress option is *enabled*, the compress flag will be *disabled* when downgrading
Once only **(once_only)**	Limits Equalizer to parsing headers (and executing match rules) for only the first request of any client making multiple requests across a single TCP connection. This option is off by default: meaning that Equalizer will parse the headers of every client request.
Rewrite Redirects (HTTPS Only) **(rewrite_redirects)**	When enabled, forces Equalizer to pass responses from an HTTPS cluster's servers without rewriting them. In the typical Equalizer setup, you configure servers in an HTTPS cluster to listen and respond using HTTP; Equalizer communicates with the clients using SSL. If a server sends an HTTP redirect using the Location: header, this URL most likely will not include the https: protocol. Equalizer rewrites responses from the server so that they are HTTPS. You can direct Equalizer to pass responses from the server without rewriting them by enabling this option.
Ignore case **(ignore_case)**	Applies to L7 clusters and is the global setting to ignore case in match expressions. You can override this value per cluster and per match rule .
Spoof **(spoof)**	When the spoof option is enabled on a cluster, Equalizer uses the client's IP address as the source IP address in all packets sent to a server in that cluster. This option is enabled by default. When spoof is enabled, all server responses to client requests that came through the Equalizer cluster IP address must be routed by the server back to the client through Equalizer. In many cases, the easiest way to do this is to set the default gateway on the server with a server instance in a server pool to Equalizer's IP address on the server VLAN. If this is not possible, you can establish static routes on the server to send responses to specific client IP addresses to Equalizer's IP address on the VLAN. If you disable spoof, the server receiving the request will see Equalizer's IP address as the client address because the TCP connection to the client is terminated when the request is routed. The server will therefore send its response back to Equalizer's IP address. Disabling the spoof option enables Source Network Address Translation (SNAT).

GUI Parameter (CLI Parameter)	Description
Ignore Critical Extensions (HTTPS only- not shown above) **(ignore_critical_extns)**	Control whether Equalizer will process "CRL Distribution Point" extensions in client certificates. This option only affects the processing of the "CRL Distribution Point" extension in client certificates: When **Ignore Critical Extensions** is disabled, a client certificate presented to Equalizer that includes any extension will be rejected by Equalizer . This is the behavior in previous releases. When **Ignore Critical Extensions** is enabled (the default), a client certificate presented to Equalizer that has a CRL Distribution Point extension will be processed and the CRL critical extension will be ignored. Note, however, that if other extensions are present in a client certificate they are not ignored and will cause the client certificate to be rejected by Equalizer.
Server Side Encryption **(sse)**	Enables the **Server Side Encryption** option for the Cluster. By enabling this option, Equalizer will encrypt packets to back end servers using the SSL/TLS specifications in the Server Side Encryption Global Parameters (See "Server Side Encryption" on page 276)

GUI Parameter (CLI Parameter)	Description
Protocol **(proto)**	The protocol selected in the Add Cluster form will be displayed “grayed out”.
VID **(vid)**	The VLAN ID number assigned to the VLAN on which the cluster resides. Refer to Common Networking Scenarios for details.
IP **(ip)**	Enter the IP address, which is the dotted decimal IP address of the cluster. The IP address of the cluster is the external address (for example, 172.16.0.201) with which clients connect to the cluster.
Port **(port)**	For HTTP and HTTPS protocol clusters, enter the port: the numeric port number on the Equalizer to be used for traffic between the clients and the cluster. For HTTP clusters, the port is normally 80. For HTTPS clusters, the port is normally 443. This port also becomes the default port for servers added to the cluster (though servers can use a different port number than the one used by the cluster).
Preferred Peer **(preferred_peer)**	Used with Active-Active Failover. Refer to Configuring Active/Active Failover Between Two Systems.
Server Pool **(srvpool)**	A drop down list used to select a Server Pool, or grouping of servers, to which the cluster will communicate with.
Responder **(resp)**	A Responder is a server-like object that can be associated with a Match Rule. If an incoming request satisfies a Match Rule expression and all of the servers specified in the Match Rule are down, a Responder definition in the Match Rule (if present) tells Equalizerto send one of two automatic responses to the client.
Custom Header **(custhdr)**	A custom HTTP header that Equalizerinserts into all client requests before they are sent to the server. The format of the string is text:text. Also see Specifying a Custom Header for HTTP/HTTPS Clusters.

GUI Parameter (CLI Parameter)	Description
Compression Minimum Size (ADCs with Hardware Acceleration) (`compress_min`)	The minimum file size in bytes required for GZIP compression, if enabled. Equalizeruses GZIP to compress the payload (or content) of the server response before sending it back to the client. This is typically done for 2 reasons: faster client response and better user experience. In addition. less ISP bandwidth is used in sending smaller files back to clients. Files smaller than the minimum specified are not compressed. Default:1024 bytes.
Compression MIME Types (ADCs with Hardware Acceleration) (`compress_types`)	Specifies the mime-types that will be compressed when the Compress option is enabled for the cluster. The value of this parameter is a string (maximum length: 512 characters) with valid mime-type names separated by a colon (:). The default compress mime-types string specifies the following mime-types text/* application/msword application/postscript application/rtf application/x-csh application/x-javascript application/x-sh application/x-shar application/x-tar application/x-tcl application/xslt+xml audio/midi audio/32kadpcm audio/x-wav image/bmp image/tiff image/x-rgb
Config Header Edit Script (`hdredit`)	When you create header edits, you place the search and edit functions within one of two meta-functions (**client_request_edit** or **server_ response_edit**) the script will be displayed here.
Flags	
Abort server (`abort_server`)	By default, when a client closes a connection, Equalizerwaits for a response from the server before closing the server connection. If this flag is enabled, Equalizer will not wait for a response before closing the connection to the server; instead it sends a TCP RST (reset) to the server when the client closes the connection. This option will typically reduce the number of server connections in the TIME_WAIT state, as shown by the netstat console command.
Insert client IP (`insert_client_ip`)	When this flag is enabled, Equalizer inserts an X-forwarded-for: header with the client's IP address into all client requests before they are sent to the server. This flag is disabled by default for HTTP clusters and enabled by default for HTTPS clusters.
TCP Multiplexing (`tcp_mux`)	This selection enables TCP multiplexing for a cluster. TCP multiplexing must also be enabled on at least one server instance in the server pool assigned to the cluster (or one of its match rules).

GUI Parameter (CLI Parameter)	Description
Allow Multibyte Characters **(`allow_utf8`)**	By default, support for extended characters (8-bit ASCII and multibyte UTF characters) in URIs is disabled. Equalizer returns a 400 Bad Request error when a request URI contains 8-bit or multibyte characters. To enable support for 8-bit and multibyte characters in URIs, click this checkbox. There are potential risks to enabling this option, because it allows Equalizer to pass requests that violate RFC2396; load-balanced servers may be running software that is incapable of handling such requests. Therefore, ensure that your server software is capable of handling URIs containing extended characters and will not serve as a potential weak point in your network before you enable extended characters.
Compress (Not applicable to E250GX) **(`compress`)**	When this option is enabled, Equalizer automatically detects requests to the cluster from compression-capable browser clients and performs GZIP compression on all cluster responses sent to that client. The cluster IP address will not accept requests when this flag is enabled. When enabled, hardware compression will be used on ADCs with compression hardware on board and software compression will be used on ADCs without compression hardware. To view the compression type used on your appliance, do one of the following: 1. At the CLI prompt, enter; **eqcli > version** . The compression type used is displayed beside **Features.** 2. In the GUI, click on the **System** configuration tab, expand the **Global** branch and click on **Dashboard**. The compression type will be displayed beside **Features** in the **System Information** widget on the right.
Once only **(`once_only`)**	Limits Equalizer to parsing headers (and executing match rules) for only the first request of any client making multiple requests across a single TCP connection. This option is off by default: meaning that Equalizer will parse the headers of every client request.
Rewrite Redirects (HTTPS Only) **(`rewrite_redirects`)**	When enabled, forces Equalizer to pass responses from an HTTPS cluster's servers without rewriting them. In the typical Equalizer setup, you configure servers in an HTTPS cluster to listen and respond using HTTP; Equalizer communicates with the clients using SSL. If a server sends an HTTP redirect using the Location: header, this URL most likely will not include the https: protocol. Equalizer rewrites responses from the server so that they are HTTPS. You can direct Equalizer to pass responses from the server without rewriting them by enabling this option.
Ignore case **(`ignore_case`)**	Applies to L7 clusters and is the global setting to ignore case in match expressions. You can override this value per cluster and per match rule .

GUI Parameter (CLI Parameter)	Description
Spoof **(spoof)**	When the spoof option is enabled on a cluster, Equalizer uses the client's IP address as the source IP address in all packets sent to a server in that cluster. This option is enabled by default. When spoof is enabled, all server responses to client requests that came through the Equalizer cluster IP address must be routed by the server back to the client through Equalizer. In many cases, the easiest way to do this is to set the default gateway on the server with a server instance in a server pool to Equalizer's IP address on the server VLAN. If this is not possible, you can establish static routes on the server to send responses to specific client IP addresses to Equalizer's IP address on the VLAN. If you disable spoof, the server receiving the request will see Equalizer's IP address as the client address because the TCP connection to the client is terminated when the request is routed. The server will therefore send its response back to Equalizer's IP address. Disabling the spoof option enables Source Network Address Translation (SNAT).
Ignore Critical Extensions (HTTPS only- not shown above) **(ignore_critical_extns)**	Control whether Equalizer will process "CRL Distribution Point" extensions in client certificates. This option only affects the processing of the "CRL Distribution Point" extension in client certificates: When **Ignore Critical Extensions** is disabled, a client certificate presented to Equalizer that includes any extension will be rejected by Equalizer . This is the behavior in previous releases. When **Ignore Critical Extensions** is enabled (the default), a client certificate presented to Equalizer that has a CRL Distribution Point extension will be processed and the CRL critical extension will be ignored. Note, however, that if other extensions are present in a client certificate they are not ignored and will cause the client certificate to be rejected by Equalizer.
Server Side Encryption **(sse)**	Enables the **Server Side Encryption** option for the Cluster. By enabling this option, Equalizer will encrypt packets to back end servers using the SSL/TLS specifications in the Server Side Encryption Global Parameters (See "Server Side Encryption" on page 276)

GUI Parameter (CLI Parameter)	Description
Protocol **(proto)**	The protocol selected in the Add Cluster form will be displayed "grayed out".
VID **(vid)**	The VLAN ID number assigned to the VLAN on which the cluster resides. Refer to Common Networking Scenarios for details.
IP **(ip)**	Enter the IP address, which is the dotted decimal IP address of the cluster. The IP address of the cluster is the external address (for example, 172.16.0.201) with which clients connect to the cluster.

GUI Parameter (CLI Parameter)	Description
Port **(port)**	For HTTP and HTTPS protocol clusters, enter the port: the numeric port number on the Equalizer to be used for traffic between the clients and the cluster. For HTTP clusters, the port is normally 80. For HTTPS clusters, the port is normally 443. This port also becomes the default port for servers added to the cluster (though servers can use a different port number than the one used by the cluster).
Preferred Peer **(preferred_peer)**	Used with Active-Active Failover. Refer to Configuring Active/Active Failover Between Two Systems.
Server Pool **(srvpool)**	A drop down list used to select a Server Pool, or grouping of servers, to which the cluster will communicate with.
Responder **(resp)**	A Responder is a server-like object that can be associated with a Match Rule. If an incoming request satisfies a Match Rule expression and all of the servers specified in the Match Rule are down, a Responder definition in the Match Rule (if present) tells Equalizerto send one of two automatic responses to the client.
Custom Header **(custhdr)**	A custom HTTP header that Equalizerinserts into all client requests before they are sent to the server. The format of the string is text:text. Also see Specifying a Custom Header for HTTP/HTTPS Clusters.
Compression Minimum Size **(ADCs with Hardware Acceleration)** **(compress_min)**	The minimum file size in bytes required for GZIP compression, if enabled. Equalizeruses GZIP to compress the payload (or content) of the server response before sending it back to the client. This is typically done for 2 reasons: faster client response and better user experience. In addition. less ISP bandwidth is used in sending smaller files back to clients. Files smaller than the minimum specified are not compressed. Default:1024 bytes.
Compression MIME Types (ADCs with Hardware Acceleration) **(compress_types)**	Specifies the mime-types that will be compressed when the Compress option is enabled for the cluster. The value of this parameter is a string (maximum length: 512 characters) with valid mime-type names separated by a colon (:). The default compress mime-types string specifies the following mime-types text/* application/msword application/postscript application/rtf application/x-csh application/x-javascript application/x-sh application/x-shar application/x-tar application/x-tcl application/xslt+xml audio/midi audio/32kadpcm audio/x-wav image/bmp image/tiff image/x-rgb
Config Header Edit Script **(hdredit)**	When you create header edits, you place the search and edit functions within one of two meta-functions (**client_request_edit** or **server_ response_edit**) the script will be displayed here.

GUI Parameter (CLI Parameter)	Description
Flags	
Abort server **(abort_server)**	By default, when a client closes a connection, Equalizerwaits for a response from the server before closing the server connection. If this flag is enabled, Equalizer will not wait for a response before closing the connection to the server; instead it sends a TCP RST (reset) to the server when the client closes the connection. This option will typically reduce the number of server connections in the TIME_WAIT state, as shown by the netstat console command.
Insert client IP **(insert_client_ip)**	When this flag is enabled, Equalizer inserts an X-forwarded-for: header with the client's IP address into all client requests before they are sent to the server. This flag is disabled by default for HTTP clusters and enabled by default for HTTPS clusters.
TCP Multiplexing **(tcp_mux)**	This selection enables TCP multiplexing for a cluster. TCP multiplexing must also be enabled on at least one server instance in the server pool assigned to the cluster (or one of its match rules).
Allow Multibyte Characters **(allow_utf8)**	By default, support for extended characters (8-bit ASCII and multibyte UTF characters) in URIs is disabled. Equalizer returns a 400 Bad Request error when a request URI contains 8-bit or multibyte characters. To enable support for 8-bit and multibyte characters in URIs, click this checkbox. There are potential risks to enabling this option, because it allows Equalizer to pass requests that violate RFC2396; load-balanced servers may be running software that is incapable of handling such requests. Therefore, ensure that your server software is capable of handling URIs containing extended characters and will not serve as a potential weak point in your network before you enable extended characters.
Compress (Not applicable to E250GX) **(compress)**	When this option is enabled, Equalizer automatically detects requests to the cluster from compression-capable browser clients and performs GZIP compression on all cluster responses sent to that client. The cluster IP address will not accept requests when this flag is enabled. When enabled, hardware compression will be used on ADCs with compression hardware on board and software compression will be used on ADCs without compression hardware. To view the compression type used on your appliance, do one of the following: 1. At the CLI prompt, enter; **eqcli > version** . The compression type used is displayed beside **Features.** 2. In the GUI, click on the **System** configuration tab, expand the **Global** branch and click on **Dashboard**. The compression type will be displayed beside **Features** in the **System Information** widget on the right.
Once only **(once_only)**	Limits Equalizer to parsing headers (and executing match rules) for only the first request of any client making multiple requests across a single TCP connection. This option is off by default: meaning that Equalizer will parse the headers of every client request.

GUI Parameter (CLI Parameter)	Description
Rewrite Redirects (HTTPS Only) **(rewrite_redirects)**	When enabled, forces Equalizer to pass responses from an HTTPS cluster's servers without rewriting them. In the typical Equalizer setup, you configure servers in an HTTPS cluster to listen and respond using HTTP; Equalizer communicates with the clients using SSL. If a server sends an HTTP redirect using the Location: header, this URL most likely will not include the https: protocol. Equalizer rewrites responses from the server so that they are HTTPS. You can direct Equalizer to pass responses from the server without rewriting them by enabling this option.
Ignore case **(ignore_case)**	Applies to L7 clusters and is the global setting to ignore case in match expressions. You can override this value per cluster and per match rule .
Spoof **(spoof)**	When the spoof option is enabled on a cluster, Equalizer uses the client's IP address as the source IP address in all packets sent to a server in that cluster. This option is enabled by default. When spoof is enabled, all server responses to client requests that came through the Equalizer cluster IP address must be routed by the server back to the client through Equalizer. In many cases, the easiest way to do this is to set the default gateway on the server with a server instance in a server pool to Equalizer's IP address on the server VLAN. If this is not possible, you can establish static routes on the server to send responses to specific client IP addresses to Equalizer's IP address on the VLAN. If you disable spoof, the server receiving the request will see Equalizer's IP address as the client address because the TCP connection to the client is terminated when the request is routed. The server will therefore send its response back to Equalizer's IP address. Disabling the spoof option enables Source Network Address Translation (SNAT).
Ignore Critical Extensions (HTTPS only- not shown above) **(ignore_critical_extns)**	Control whether Equalizer will process "CRL Distribution Point" extensions in client certificates. This option only affects the processing of the "CRL Distribution Point" extension in client certificates: When **Ignore Critical Extensions** is disabled, a client certificate presented to Equalizer that includes any extension will be rejected by Equalizer . This is the behavior in previous releases. When **Ignore Critical Extensions** is enabled (the default), a client certificate presented to Equalizer that has a CRL Distribution Point extension will be processed and the CRL critical extension will be ignored. Note, however, that if other extensions are present in a client certificate they are not ignored and will cause the client certificate to be rejected by Equalizer.
Server Side Encryption **(sse)**	Enables the **Server Side Encryption** option for the Cluster. By enabling this option, Equalizer will encrypt packets to back end servers using the SSL/TLS specifications in the Server Side Encryption Global Parameters (See "Server Side Encryption" on page 276)

GUI Parameter (CLI Parameter)	Description
Protocol **(proto)**	The protocol selected in the Add Cluster form will be displayed "grayed out".

GUI Parameter (CLI Parameter)	Description
VID **(vid)**	The VLAN ID number assigned to the VLAN on which the cluster resides. Refer to Common Networking Scenarios for details.
IP **(ip)**	Enter the IP address, which is the dotted decimal IP address of the cluster. The IP address of the cluster is the external address (for example, 172.16.0.201) with which clients connect to the cluster.
Port **(port)**	For HTTP and HTTPS protocol clusters, enter the port: the numeric port number on the Equalizer to be used for traffic between the clients and the cluster. For HTTP clusters, the port is normally 80. For HTTPS clusters, the port is normally 443. This port also becomes the default port for servers added to the cluster (though servers can use a different port number than the one used by the cluster).
Preferred Peer **(preferred_peer)**	Used with Active-Active Failover. Refer to Configuring Active/Active Failover Between Two Systems.
Server Pool **(srvpool)**	A drop down list used to select a Server Pool, or grouping of servers, to which the cluster will communicate with.
Responder **(resp)**	A Responder is a server-like object that can be associated with a Match Rule. If an incoming request satisfies a Match Rule expression and all of the servers specified in the Match Rule are down, a Responder definition in the Match Rule (if present) tells Equalizerto send one of two automatic responses to the client.
Custom Header **(custhdr)**	A custom HTTP header that Equalizerinserts into all client requests before they are sent to the server. The format of the string is text:text. Also see Specifying a Custom Header for HTTP/HTTPS Clusters.
Compression Minimum Size (ADCs with Hardware Acceleration) **(compress_min)**	The minimum file size in bytes required for GZIP compression, if enabled. Equalizeruses GZIP to compress the payload (or content) of the server response before sending it back to the client. This is typically done for 2 reasons: faster client response and better user experience. In addition. less ISP bandwidth is used in sending smaller files back to clients. Files smaller than the minimum specified are not compressed. Default:1024 bytes.

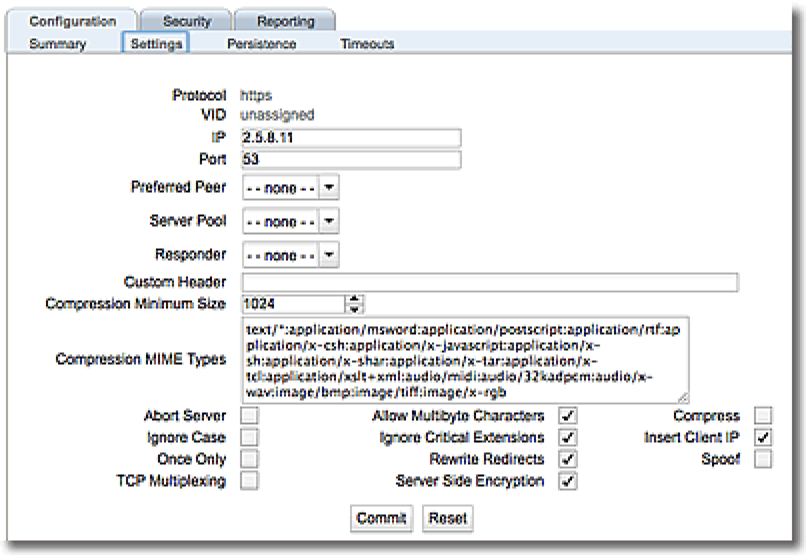
GUI Parameter (CLI Parameter)	Description
Compression MIME Types (ADCs with Hardware Acceleration) (`compress_types`)	Specifies the mime-types that will be compressed when the Compress option is enabled for the cluster. The value of this parameter is a string (maximum length: 512 characters) with valid mime-type names separated by a colon (:). The default compress mime-types string specifies the following mime-types text/* application/msword application/postscript application/rtf application/x-csh application/x-javascript application/x-sh application/x-shar application/x-tar application/x-tcl application/xslt+xml audio/midi audio/32kadpcm audio/x-wav image/bmp image/tiff image/x-rgb
Config Header Edit Script (`hdredit`)	When you create header edits, you place the search and edit functions within one of two meta-functions (**client_request_edit** or **server_ response_edit**) the script will be displayed here.
Flags	
Abort server (`abort_server`)	By default, when a client closes a connection, Equalizerwaits for a response from the server before closing the server connection. If this flag is enabled, Equalizer will not wait for a response before closing the connection to the server; instead it sends a TCP RST (reset) to the server when the client closes the connection. This option will typically reduce the number of server connections in the TIME_WAIT state, as shown by the netstat console command.
Insert client IP (`insert_client_ip`)	When this flag is enabled, Equalizer inserts an X-forwarded-for: header with the client's IP address into all client requests before they are sent to the server. This flag is disabled by default for HTTP clusters and enabled by default for HTTPS clusters.
TCP Multiplexing (`tcp_mux`)	This selection enables TCP multiplexing for a cluster. TCP multiplexing must also be enabled on at least one server instance in the server pool assigned to the cluster (or one of its match rules).
Allow Multibyte Characters (`allow_utf8`)	By default, support for extended characters (8-bit ASCII and multibyte UTF characters) in URIs is disabled. Equalizer returns a 400 Bad Request error when a request URI contains 8-bit or multibyte characters. To enable support for 8-bit and multibyte characters in URIs, click this checkbox. There are potential risks to enabling this option, because it allows Equalizer to pass requests that violate RFC2396; load-balanced servers may be running software that is incapable of handling such requests. Therefore, ensure that your server software is capable of handling URIs containing extended characters and will not serve as a potential weak point in your network before you enable extended characters.

GUI Parameter (CLI Parameter)	Description
Compress (Not applicable to E250GX) **(`compress`)**	When this option is enabled, Equalizer automatically detects requests to the cluster from compression-capable browser clients and performs GZIP compression on all cluster responses sent to that client. The cluster IP address will not accept requests when this flag is enabled. When enabled, hardware compression will be used on ADCs with compression hardware on board and software compression will be used on ADCs without compression hardware. To view the compression type used on your appliance, do one of the following: 1. At the CLI prompt, enter; **eqcli > version** . The compression type used is displayed beside **Features.** 2. In the GUI, click on the **System** configuration tab, expand the **Global** branch and click on **Dashboard**. The compression type will be displayed beside **Features** in the **System Information** widget on the right.
Once only **(`once_only`)**	Limits Equalizer to parsing headers (and executing match rules) for only the first request of any client making multiple requests across a single TCP connection. This option is off by default: meaning that Equalizer will parse the headers of every client request.
Rewrite Redirects (HTTPS Only) **(`rewrite_redirects`)**	When enabled, forces Equalizer to pass responses from an HTTPS cluster's servers without rewriting them. In the typical Equalizer setup, you configure servers in an HTTPS cluster to listen and respond using HTTP; Equalizer communicates with the clients using SSL. If a server sends an HTTP redirect using the Location: header, this URL most likely will not include the https: protocol. Equalizer rewrites responses from the server so that they are HTTPS. You can direct Equalizer to pass responses from the server without rewriting them by enabling this option.
Ignore case **(`ignore_case`)**	Applies to L7 clusters and is the global setting to ignore case in match expressions. You can override this value per cluster and per match rule .
Spoof **(`spoof`)**	When the spoof option is enabled on a cluster, Equalizer uses the client's IP address as the source IP address in all packets sent to a server in that cluster. This option is enabled by default. When spoof is enabled, all server responses to client requests that came through the Equalizer cluster IP address must be routed by the server back to the client through Equalizer. In many cases, the easiest way to do this is to set the default gateway on the server with a server instance in a server pool to Equalizer's IP address on the server VLAN. If this is not possible, you can establish static routes on the server to send responses to specific client IP addresses to Equalizer's IP address on the VLAN. If you disable spoof, the server receiving the request will see Equalizer's IP address as the client address because the TCP connection to the client is terminated when the request is routed. The server will therefore send its response back to Equalizer's IP address. Disabling the spoof option enables Source Network Address Translation (SNAT).

GUI Parameter (CLI Parameter)	Description
Ignore Critical Extensions (HTTPS only- not shown above) (`ignore_critical_extns`)	Control whether Equalizer will process "CRL Distribution Point" extensions in client certificates. This option only affects the processing of the "CRL Distribution Point" extension in client certificates: When **Ignore Critical Extensions** is disabled, a client certificate presented to Equalizer that includes any extension will be rejected by Equalizer . This is the behavior in previous releases. When **Ignore Critical Extensions** is enabled (the default), a client certificate presented to Equalizer that has a CRL Distribution Point extension will be processed and the CRL critical extension will be ignored. Note, however, that if other extensions are present in a client certificate they are not ignored and will cause the client certificate to be rejected by Equalizer.
Server Side Encryption (`sse`)	Enables the **Server Side Encryption** option for the Cluster. By enabling this option, Equalizer will encrypt packets to back end servers using the SSL/TLS specifications in the Server Side Encryption Global Parameters (See "Server Side Encryption" on page 276)

Configuring Layer 7 HTTP or HTTPS Clusters Using the GUI

The Layer 7 Cluster Configuraiton Screen, as shown below, is displayed whey you select Load Balance > Clusters and select a cluster from the branch. When you click on the Configuration tab, the folowing will be displayed.



Protocol	https
VID	unassigned
IP	2.5.8.11
Port	53
Preferred Peer	- - none - -
Server Pool	- - none - -
Responder	- - none - -
Custom Header	
Compression Minimum Size	1024
Compression MIME Types	text/*:application/msword:application/postscript:application/rtf:application/x-csh:application/x-javascript:application/x-sh:application/x-shar:application/x-tar:application/x-tcl:application/xslt+xml:audio/midi:audio/32kadpcm:audio/x-wav:image/bmp:image/tiff:image/x-rgb

Abort Server	☐	Allow Multibyte Characters	✓	Compress	☐
Ignore Case	☐	Ignore Critical Extensions	✓	Insert Client IP	✓
Once Only	☐	Rewrite Redirects	✓	Spoof	☐
TCP Multiplexing	☐	Server Side Encryption	✓		

Use the table above for parameters, values, and flags for the configuration of an HTTP or HTTPS cluster. Click on the **Commit** button after making changes to the settings.

Configuring Layer 7 HTTP or HTTPS Clusters Using the CLI

Layer 7 HTTP and HTTPS Clusters can be configured in the CLI either globally or in cluster context. Enter parameters using the following format.

```
eqcli > cluster clustername parameter value flags flag
```

Use the table above for descriptions of the parameters and values.

Where:

clustername - is the the name fo the cluster.

parameter - is the parameter.

value - is the value associated with the parameter.

flag - is the flag to be associated with the cluster.

Use the table above for parameters and values .

Refer to "Cluster and Match Rule Commands" on page 171 for additional information on using cluster commands in the CLI.

Layer 7 Security Certificate Screen (HTTPS Clusters)

The HTTPS protocol supports encrypted, secure communication between clients and servers. It requires that a Secure Sockets Layer (SSL) authentication handshake occur between a client and a server in order for a connection request to succeed.

Certificates are loaded using either the GUI or the CLI.

Parameters

The table below shows the parameters, values, and flags that are used in the configuration of Layer 7 HTTPS Cluster security.

GUI Parameter (CLI Parameter)	Descriptions
Default Certificate **(`certificate`)**	Use the drop down list to select a default SSL certificate that clients will use to validate a connection to this HTTPS cluster.
Client CA **(`clientca`)**	The **Client CA** is used to authenticate the SSL client certificate if the **Require Client Certificate** option is enabled or if a **CRL** selection is made. Use the drop down list to select the name of a client certificate authority (CA).This is the certificate of an authority in a network that issues and manages security credentials and public keys for message encryption. It must be uploaded to Equalizer's certificated store. As part of a public key infrastructure, a CA checks with a registration authority to verify information provided by the requester of a digital certificate. If the registration authority verifies the requester's information, the CA can then issue a certificate. The certificate usually includes the owner's public key, the expiration date of the certificate, the owner's name, and other information about the public key owner.
CRL **(`crl`)**	A Certificate Revocation List **CRL** is used to check if the SSL certificates provided by the SSL client during the SSL handshake are not in the CRL list. It requires the **Client CA** to be specified. Use the drop down list to select a **CRL**.
Validation Depth **(`valdepth`)**	The depth to which certificate checking is done on the client certificate chain. The default of 2 indicates that the client certificate (level 0) and two levels above it (levels 1 and 2) are checked; any certificates above level 2 in the chain are ignored. You should only need to increase this value if the Certificate Authority that issued your certificate provided you with more than 2 chained certificates in addition to your client certificate.
Flags	
Push Client Certificate **(`push_client_cert`)**	Enabling this option sends the client certificate to the back-end server.
Require Client Certificate **(`require_client_cert`)**	Enabling this option requires that client's present certificates. The client CA, if configured, validates the SSL certificate presented by the SSL client.

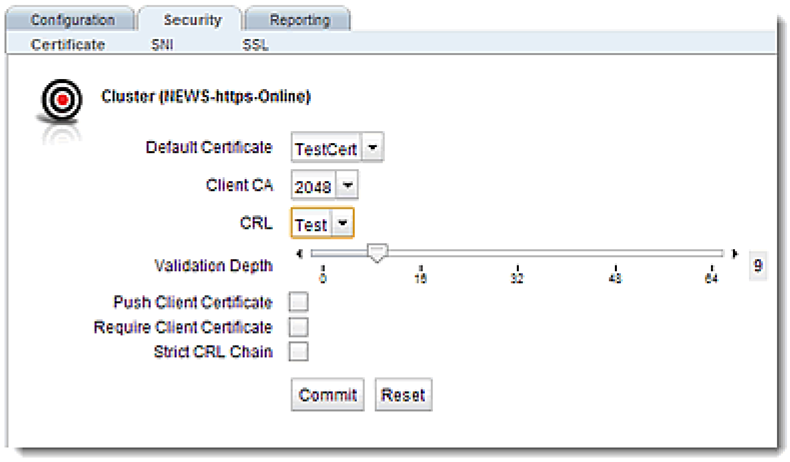
GUI Parameter (CLI Parameter)	Descriptions
Strict CRL Chain (**`strict_crl_chain`**)	This option requires the **Client CA** and **CRL** to be specified. If it is enabled then it ensures that none of the certificates in the certificate chain of the SSL client certificate are in the **CRL**. If the client **CA** and **CRL** are specified, yet this option is not enabled, then only the last certificate in the certificate chain of the SSL client certificate is checked against the specified **CRL**.

GUI Parameter (CLI Parameter)	Descriptions
Default Certificate (**`certificate`**)	Use the drop down list to select a default SSL certificate that clients will use to validate a connection to this HTTPS cluster.
Client CA (**`clientca`**)	The **Client CA** is used to authenticate the SSL client certificate if the **Require Client Certificate** option is enabled or if a **CRL** selection is made. Use the drop down list to select the name of a client certificate authority (CA).This is the certificate of an authority in a network that issues and manages security credentials and public keys for message encryption. It must be uploaded to Equalizer's certificated store. As part of a public key infrastructure, a CA checks with a registration authority to verify information provided by the requester of a digital certificate. If the registration authority verifies the requester's information, the CA can then issue a certificate. The certificate usually includes the owner's public key, the expiration date of the certificate, the owner's name, and other information about the public key owner.
CRL (**`crl`**)	A Certificate Revocation List **CRL** is used to check if the SSL certificates provided by the SSL client during the SSL handshake are not in the CRL list. It requires the **Client CA** to be specified. Use the drop down list to select a **CRL**.
Validation Depth (**`valdepth`**)	The depth to which certificate checking is done on the client certificate chain. The default of 2 indicates that the client certificate (level 0) and two levels above it (levels 1 and 2) are checked; any certificates above level 2 in the chain are ignored. You should only need to increase this value if the Certificate Authority that issued your certificate provided you with more than 2 chained certificates in addition to your client certificate.
Flags	
Push Client Certificate (**`push_client_cert`**)	Enabling this option sends the client certificate to the back-end server.
Require Client Certificate (**`require_client_cert`**)	Enabling this option requires that client's present certificates. The client CA, if configured, validates the SSL certificate presented by the SSL client.
Strict CRL Chain (**`strict_crl_chain`**)	This option requires the **Client CA** and **CRL** to be specified. If it is enabled then it ensures that none of the certificates in the certificate chain of the SSL client certificate are in the **CRL**. If the client **CA** and **CRL** are specified, yet this option is not enabled, then only the last certificate in the certificate chain of the SSL client certificate is checked against the specified **CRL**.

If you have uploaded a certificate that doesn't match the cipher suite that is configured for the HTTPS cluster, you will no longer be able to log into the GUI. You will need to supply the correct certificate/key pairing. In the meantime, you can enable HTTP access to the GUI temporarily to enter the proper certificate/key pairing to enable HTTPS access.

Loading Certificates Using the GUI

The Layer 7 **Security > Certificate** screen shown below is available when an HTTPS cluster is selected from the **Cluster** branch on the left navigational pane.



Cluster (NEWS-https-Online)

Default Certificate: TestCert
Client CA: 2048
CRL: Test
Validation Depth: 9
Push Client Certificate: ☐
Require Client Certificate: ☐
Strict CRL Chain: ☐

Use the **Security > Certificate** tab to select a default SSL certificate that clients will use to validate a connection to an HTTPS cluster (a cluster certificate).

Use the table above for parameters, values, and flags for the configuration of HTTPS cluster certificates. Click on the **Commit** button after making changes to the settings.

Loading Certificates Using the CLI

Layer 7 HTTPS cluster certificates can be configured in the CLI either globally or in cluster context. Enter parameters using the following format.

```
eqcli > cluster clustername certificate certificate name flags flag
```

Use the table above for descriptions of the parameters and values. The certicate commands can only be entered if the protocol (**`proto`**) is **`https`**.

`clustername` - is the the name fo the cluster.

`certificate name` - is the name of the certificate being uploaded.

`flag` - is the flag to be associated with the cluster.

Use the table above for parameters and values .

Refer to "Cluster and Match Rule Commands" on page 171 for additional information on using cluster commands in the CLI.

Layer 7 SSL Security (HTTPS Clusters)

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide secure communications across HTTPS clusters.

A cipher is an algorithm that performs encryption or decryption. It transforms plain text into a coded set of data (cipher text) that is not reversible without a key. For example, AES and DES are examples of secret key block ciphers. The complete encryption algorithm is the cipher plus the protocol used to apply the cipher to the message.

The **Layer 7 SSL Security** screen is used to configure TLS/SSL protocol and ciphers that are used by Equalizer to communicate with clients over HTTPS clusters.

The **Cipher Suites** for an HTTPS cluster lists all of the ciphers that can be negotiated between Equalizer and an incoming client attempting to connect to an HTTPS cluster. Similarly, the client application will have its own list of ciphers that it supports. The client and Equalizer need to go through a process of negotiating the cipher that will be used for the client connection -- if they cannot find a match, the connection will fail. The process of negotiating a cipher for a client connection is as follows:

1. During the SSL handshake phase of the connection, the client sends Equalizer a list of the ciphers it supports.
2. Equalizer examines the client cipher list in the order it is specified, chooses the first cipher that matches a cipher specified in the cluster’s cipher suite parameter, and responds to the client. If none of the ciphers offered by the client are in the cipher suite list for the cluster, the SSL handshake fails.

It is therefore vital that you ensure that there is at least one match between the list of ciphers supported by clients connecting to an HTTPS cluster and the cipher suite list for the cluster.

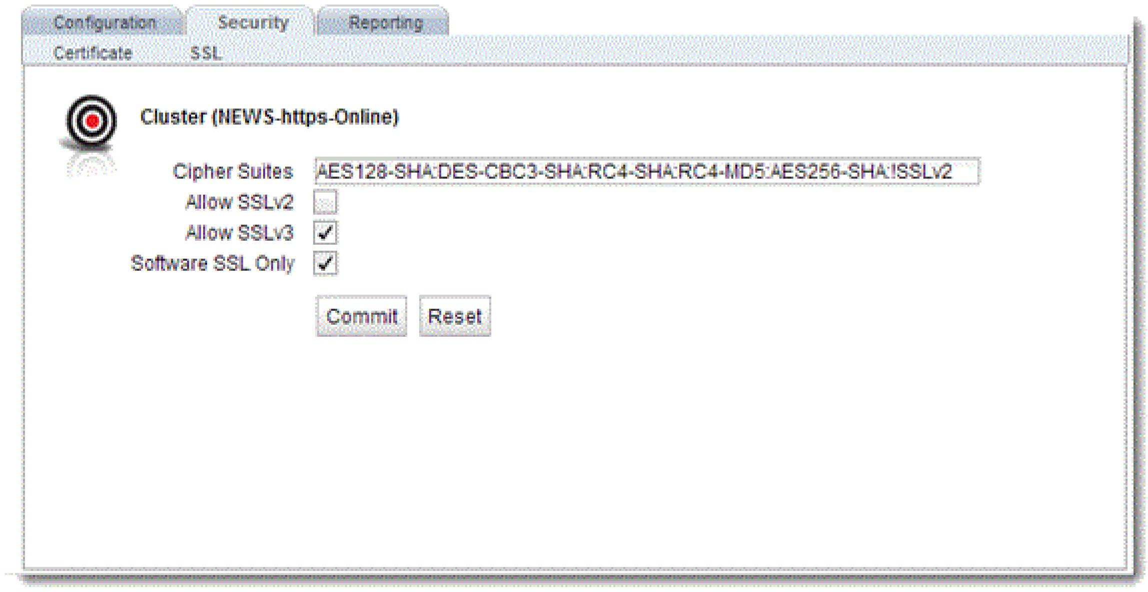
Parameters

The table below shows the parameters and values used in the configuration of HTTPS cluster security.

GUI Parameter (CLI Parameter)	Description
Cipher Suites (`cipherspec`)	The list of ciphers supported for encrypted cluster connections. The order of the suites in the list indicates the order in which a cipher is selected for the incoming connection. During the SSL handshake phase of the connection, the client sends a list of the ciphers it supports. The ADC chooses the first cipher in the cluster's configured cipher list that matches a cipher specified by the client. If none of the ciphers offered by the client are in the cluster's configured cipher list, the SSL handshake fails. For a full list of supported ciphers, see "Supported Software Cipher Suites" on page 914.
Flags	
Allow SSLv2 (`allow_sslv2`)	Enables SSLv2 for client connections.

GUI Parameter (CLI Parameter)	Description
Allow SSLv3 **(`allow_sslv3`)**	Enables SSLv3 for client connections. This option is enabled by default.
Software SSL Only **(`software_ssl_only`)** **(not applicable on E250GX)**	This flag appears only on systems that are equipped with Hardware SSL Acceleration. When enabled, it specifies that all SSL operations will be performed in software, instead of being performed using the SSL accelerator hardware. This flag does not appear on systems that are not equipped with Hardware SSL Acceleration, since on these units SSL operations are always performed in software. This flag is disabled by default. All units with Hardware SSL Acceleration can process the TLSv1.0, TLSv1.1, and TLSv1.2 protocols in both hardware and software, except for legacy GX hardware. On legacy GX hardware, only TLSv1.0 is supported by Hardware SSL Acceleration; if you want to enable TLSv1.1 or TLSv1.2 on GX hardware, you must first enable this flag. Please note that enabling this option will reduce the processor and memory resources generally available for processing cluster traffic, since performing SSL operations in software requires use of the system CPU and system memory (instead of the dedicated SSL acceleration hardware CPU and memory).
Allow TLS 1.0 **(`allow_tls10`)**	This option enables and disables support for the TLSv1.0 protocol. Enabled by default. If multiple TLS versions are enabled, the first supported TLS version negotiated by a client will be used.
Allow TLS 1.1 **(`allow_tls11`)**	This option enables and disables support for the TLSv1.1 protocol. Disabled by default. If multiple TLS versions are enabled, the first supported TLS version negotiated by a client will be used.
Allow TLS 1.2 **(`allow_tls12`)**	This option enables and disables support for the TLSv1.2 protocol. Disabled by default. If multiple TLS versions are enabled, the first supported TLS version negotiated by a client will be used.

The Layer 7 Security SSL screen shown below is displayed when an HTTPS cluster is selected from the **Cluster** branch on the left navigational pane on the GUI.



Cluster (NEWS-https-Online)

Cipher Suites AES128-SHA:DES-CBC3-SHA:RC4-SHA:RC4-MD5:AES256-SHA:!SSLv2

Allow SSLv2 ☐

Allow SSLv3 ✓

Software SSL Only ✓

Use the table above for parameters, values, and flags for the SSL configuration of an HTTPS cluster. Click on the **Commit** button after making changes to the settings.

Layer 7 HTTPS Security can be configured in the CLI either globally or in cluster context. Enter parameters using the following format.

```
eqcli > cluster clustername parameter value flags flag
```

Use the table above for descriptions of the parameters and values.

Where:

clustername - is the the name fo the cluster.

parameter - is the parameter.

value - is the value associated with the parameter.

flag - is the flag to be associated with the cluster.

Use the table above for parameters and values .

Refer to "Cluster and Match Rule Commands" on page 171 for additional information on using cluster commands in the CLI.

Layer 7 HTTP and HTTPS Cluster Persistence

Equalizer can use cookies or a server’s IP address to maintain a persistent session between a client and a particular server. A cookie is included with the server’s response header on its way back to the client. This cookie uniquely identifies the server to which the client was just connected. Equalizer routes the first request from a client using load balancing criteria; subsequent client requests are routed to the same selected server for the entire session (while the cookie is valid -- see Cookie Age, above). In that way, a server’s IP address can alternately be embedded into a response header that identifies the server.

You can also configure both cookie and source IP persistence on a cluster, which provides "fallback persistence". Fallback persistence allows you to provide a secondary method of persistence to use, in case the first method fails. For example, let's say you have Cookie persistence followed by Source IP persistence defined on a cluster. Let's also say that the client's browser has been configured by the user to block cookies received from websites. When the client sends its first request, a cookie is added to the request by the ADC before it is sent to the server. The server then responds back, and that cookie goes back to the client in the server response. When the client sends its next request to the cluster, the client browser strips the cookie from the request. When the client request is received by the cluster, it expects there to be a cookie, but there is none. Because the cluster is configured with fallback persistence, it will now use Source IP persistence instead.

Persistence options on an HTTP or HTTPS cluster are configured using either the GUI or the CLI.

Persistence Parameters

The table below shows the parameters, values, and flags used in the configuration of HTTP and HTTPS cluster persistence.

Cookie Type	**Description**
Cookie 0:Cluster IP/Port, Server IP/Port	Constructs a cookie which will be named in such a way that so as long as the cluster maintains the same IP address, servers can be added to and removed from the cluster without invalidating all of the existing cookies. **This cookie stores the cluster IP and port, and the server IP and port.**
Cookie 1:Cluster IP, Server IP /Port	Constructs a cookie which will be valid across all clusters with the same IP address (not port specific). A requirement for this to be useful is that all clusters on that IP address share the same set of servers. **This cookie stores the Cluster IP, and Server IP and port.**
Cookie 2:Cluster IP, Server IP	Constructs a cookie which will be valid across all clusters with the same IP address (using any port), and the same server within those clusters (with the server using any port). A requirement for this to be useful is that all clusters on that IP address share the same set of servers. **This cookie encodes the Cluster IP and Server IP.**

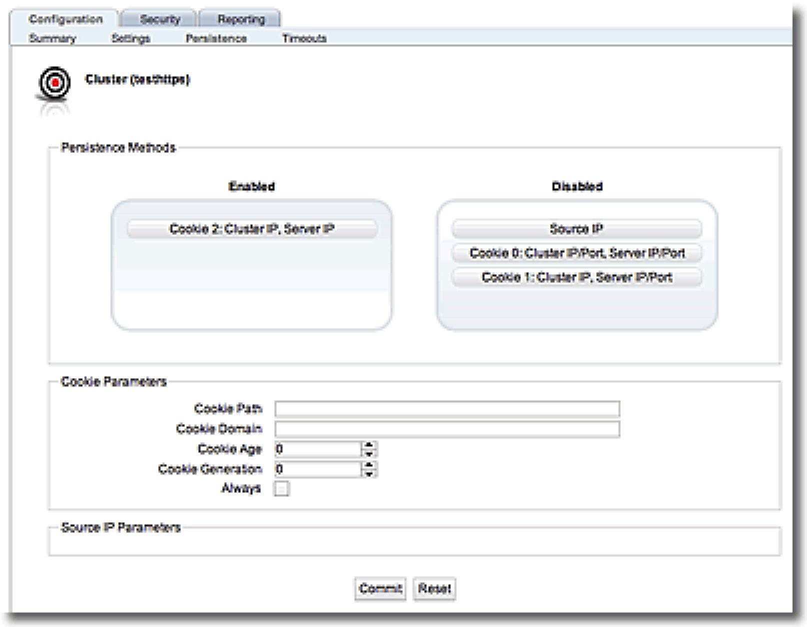
Cookie Type	Description
Source IP	The Source IP address of the server will be embedded in the response header back to the client.
Cookie Parameters **(The Cookie Parameters pane will expand if a cookie scheme is enabled.)**	
Cookie Parameter	**Description**
Cookie age	The **Cookie age** sets the time, in seconds, over which the client browser maintains the cookie ("0" means the cookie never expires). After the specified number of seconds have elapsed, the browser deletes the cookie and any subsequent client requests will be handled by Equalizer's load-balancing algorithms
Cookie path	If a **Cookie Path** is specified, Equalizer honors cookies in a client requests only when the path component of the request URI has the same prefix as that of the specified **Cookie Path**. For example, if the cookie path is /store/, Equalizer presents the cookie to the server only if the request URI includes a path such as /store/ mypage.html.
Cookie Domain -	If a cookie domain is specified, then Equalizer will honor cookies in client requests only if the server's host name is within the specified domain. For example, if the cookie domain is website.com, then Equalizer will only present the cookie to servers in the website.com domain (for example www.website.com). Wildcards are not supported in the cookie domain.
Cookie Generation -	A value added to cookies when the cookie scheme is 2. In order for cookies to be valid, the specified **Cookie Generation** must match the equivalent number embedded in the cookie. Conversely, if you need to invalidate old cookies, increment this number. **Always** - When this flag is disabled Equalizer will insert a cookie if a server was not selected based on a cookie received from the client. A cookie would only be inserted when a new client is seen or if cookie is received or if a cookie received cannot validate a server. If the **Always** flag is enabled, Equalizer includes a cookie in the response regardless of whether the server sent a cookie.
Source IP Parameters **(The Source IP pane will expand if Source IP is moved to the Enabled pane.)**	
Sticky Timeout	The number of seconds that Equalizer should "remember" connections from clients) Valid values are from 0 (which disables sticky connections) to 1073741823 seconds (or over 34 years).

Cookie Type	Description
Sticky Netmask	Enables sticky network aggregation for a subnet. Sticky network aggregation is applicable for Layer 4 and Layer 7 clusters. Sticky network aggregation enables Equalizer to correctly handle sticky connections from ISPs that use multiple proxy servers to direct user connections. When you enable sticky network aggregation, all the connections coming from a particular network are directed to the same server. (Typically, all the servers in a proxy farm are on the same network.) The sticky netmask value indicates which portion of the address Equalizer should use to identify particular networks. Values are: **0-32** for IPV4 clusters (default=32) **0-128** for IPV6 clusters
Inter cluster Sticky	With the inter-cluster sticky option, you can configure Equalizer to direct requests from a client to the same server on any available port that has a current persistent connection in any cluster. Inter-Cluster Sticky is a Layer 4 or Layer7 option that allows you to extend Layer 4 or Layer 7 persistence across multiple server ports.

Note - Inter-Cluster Sticky does not work for stickiness between a Layer 4 and a Layer 7 cluster only between a Layer 4/Layer 4 cluster or a Layer7/Layer 7 cluster.

Note - If you are using two Equalizer in a failover configuration, you must set the sticky network aggregation mask identically on both Equalizers.

Selecting a cluster from the left navigational pane on the GUI and selecting the **Persistence** tab to display the screen shown below.



Cluster (testhttps)

Persistence Methods

Enabled: Cookie 2: Cluster IP, Server IP

Disabled: Source IP; Cookie 0: Cluster IP/Port, Server IP/Port; Cookie 1: Cluster IP, Server IP/Port

Cookie Parameters

Cookie Path
Cookie Domain
Cookie Age 0
Cookie Generation 0
Always ☐

Source IP Parameters

There are three configuration panes on this screen. **Persistence Methods**, **Cookie Parameters and Source IP Parameters.**

Persistence Methods

With the **Persistence Methods** pane are an **Enabled** area and a **Not Used** area. One Persistence Type method and one Fallback Persistence Type only can be enabled. Enable and order persistence methods by dragging and dropping from between the **Not Used** area and the **Enable** area. Arrange the order between the primary persistence method and the "fallback" persistence method by dragging and dropping as well. As indicated previously, with "fallback persistence" Equalizer provides a secondary option where if, for example, a cookie response is not received, a secondary, or "fallback" option such as **Source IP** can be used.

The cookie scheme specifies the format of the cookie to be used for the cluster as an integer between 0 and 2 (default is 2).

Sticky Record Behavior

"Sticky" connections are managed on Equalizer using "Sticky Records" that record the IP address, port and other information for the client-server connection. When you enable sticky connections, the memory and CPU overhead for a connection increase. Sticky records are deleted as follows:

1. If cluster is deleted, the IP changes, or the port changes-- all source IP sticky records for that cluster are deleted.
2. If server is deleted -- all source IP sticky records for the server are deleted.
3. If server is marked down -- all source IP sticky records for the server are deleted .
4. If server instance weight is set to "0" -- all source IP sticky records for the server are deleted.
5. If server instance is marked "DOWN" -- all source IP sticky records for the server are deleted.

To verify that sticky records have been removed after any of the above scenarios were done, view the statistics in either the CLI or the GUI and observe the **CURRSTKY** (**Current Sticky Records** in the GUI) statistic to verify that the sticky records value is "0".

The following is an example of an https cluster using the CLI:

```
eqcli > cluster cl-https stats
                    Current     60 sec      10 min      60 min
TOTALPRCSD          3           0.02        0.01        0.01
TOTALRESPPRCSD      3           0.02        0.01        0.01
TIMESPENT           78          N/A         N/A         N/A
ACTIVECONX          1           0.17        0.05        0.05
BYTERCVD            97          N/A         N/A         N/A
BYTESEND            993         N/A         N/A         N/A
DROPNOSRVR          0           N/A         N/A         N/A
TOTALSTKY           0           N/A         N/A         N/A
CURRSTKY            1           0.17        0.17        0.17
```

Change the Cluster IP from 172.16.165.11 to 172.16.165.12:

```
eqcli > cluster cl-https ip 172.16.165.12
eqcli: 12000287: Operation successful
eqcli > cluster cl-https stats
                    Current     60 sec      10 min      60 min
TOTALPRCSD          3           0.02        0.00        0.00
TOTALRESPPRCSD      3           0.02        0.00        0.00
TIMESPENT           78          N/A         N/A         N/A
ACTIVECONX          0           0.17        0.05        0.05
BYTERCVD            97          N/A         N/A         N/A
BYTESEND            993         N/A         N/A         N/A
DROPNOSRVR          0           N/A         N/A         N/A
TOTALSTKY           0           N/A         N/A         N/A
CURRSTKY            0           0.33        0.19        0.19
```

Fallback Persistence Scenarios

The table below shows all of the possible persistence scenarios and the resulting load balancing server selections based on the persist types and fallback persist types that are "enabled".

Persist Type	Fallback Persist Type	Result
[none]	**[none]**	The server is selected on the load balancing Policy/Algorithm.
[none]	**Source IP**	invalid configuration
[none]	**Cookie 0:Cluster IP/Port, Server IP/Port**	invalid configuration
[none]	**Cookie 1:Cluster IP, Server IP /Port**	invalid configuration
[none]	**Cookie 2:Cluster IP, Server IP**	invalid configuration
Source IP	**[none]**	A server is selected on a sticky record (Source IP). If no records are found a server is selected and a new sticky record is created.
Source IP	**Source IP**	invalid configuration
Source IP	**Cookie 0:Cluster IP/Port, Server IP/Port**	A server is selected on a sticky record(Source IP). If no records are found a server is selected on the basis of the cookie and if the cookie is anything other than Cookie 0:Cluster IP/Port, Server IP/Port a server is selected using the Load balancing Policy/Algorithm.
Source IP	**Cookie 1:Cluster IP, Server IP /Port**	A server is selected on a sticky record(Source IP). If no records are found a server is selected on the basis of the cookie and if the cookie is anything other than Cookie 1:Cluster IP, Server IP /Port a server is selected using the Load balancing Policy/Algorithm.
Source IP	**Cookie 2:Cluster IP, Server IP**	A server is selected on a sticky record(Source IP). If no records are found a server is selected on the basis of the cookie and if the cookie is anything other than Cookie 2:Cluster IP, Server IP a server is selected using the Load balancing Policy/Algorithm.
Cookie 0:Cluster IP/Port, Server IP/Port	**[none]**	A server is selected based on the cookie If no cookie or a cookie other then Cookie 0:Cluster IP/Port, Server IP/Port is in the request the server is selected using the Load balancing Policy/Algorithm.

Persist Type	Fallback Persist Type	Result
Cookie 0:Cluster IP/Port, Server IP/Port	**Source IP**	A server is selected based on the cookie. If no cookie or a cookie other then Cookie 0:Cluster IP/Port, Server IP/Port is in the request, a server is selected on the basis of the sticky record(Source IP). If no records are found a server is selected and a new sticky record is created.
Cookie 0:Cluster IP/Port, Server IP/Port	**Cookie 0:Cluster IP/Port, Server IP/Port**	invalid configuration
Cookie 0:Cluster IP/Port, Server IP/Port	**Cookie 1:Cluster IP, Server IP /Port**	A server is selected based on the cookie. If no cookie or a cookie other then coyote_cooke_0 or Cookie 1:Cluster IP, Server IP /Port is in the request the server is selected using the Load balancing Policy/Algorithm.
Cookie 0:Cluster IP/Port, Server IP/Port	**Cookie 2:Cluster IP, Server IP**	A server is selected based on the cookie. If no cookie or a cookie other then Cookie 0:Cluster IP/Port, Server IP/Port or Cookie 2:Cluster IP, Server IP is in the request the server is selected using the Load balancing Policy/Algorithm.
Cookie 1:Cluster IP, Server IP /Port	**[[none]]**	A server is selected based on the cookie. If no cookie is in the request the server is selected using the Load balancing Policy/Algorithm.
Cookie 1:Cluster IP, Server IP /Port	**Source IP**	A server is selected based on the cookie. If no cookie or a cookie other then Cookie 1:Cluster IP, Server IP /Port is in the request, a server is selected based on the sticky record(Source IP). If no records are found a server is selected and a new sticky record is created.
Cookie 1:Cluster IP, Server IP /Port	**Cookie 0:Cluster IP/Port, Server IP/Port**	A server is selected based on the cookie. If no cookie or a cookie other then Cookie 1:Cluster IP, Server IP /Port, Server IP/Port or Cookie 0:Cluster IP/Port, Server IP/Port is in the request a server is selected using the Load balancing Policy/Algorithm.
Cookie 1:Cluster IP, Server IP /Port	**Cookie 1:Cluster IP, Server IP /Port**	invalid configuration
Cookie 1:Cluster IP, Server IP /Port	**Cookie 2:Cluster IP, Server IP**	A server is selected based on the cookie. If no cookie or a cookie other then Cookie 1:Cluster IP, Server IP /Port or Cookie 2:Cluster IP, Server IP is in the request a server is selected using the Load balancing Policy/Algorithm.
Cookie 2:Cluster IP, Server IP	**[none]**	A server is selected based on the cookie. If no cookie is in the request a server is selected using the Load balancing Policy/Algorithm.

Persist Type	Fallback Persist Type	Result
Cookie 2:Cluster IP, Server IP	**Source IP**	A server is selected based on the cookie. If no cookie or a cookie other then Cookie 2:Cluster IP, Server IP is in the request, a server is selected based on the on sticky record(Source IP). If no records are found a server is selected and a new sticky record is created.
Cookie 2:Cluster IP, Server IP	**Cookie 0:Cluster IP/Port, Server IP/Port**	A server is selected based on the cookie. If no cookie or a cookie other then Cookie 2:Cluster IP, Server IP or Cookie 0:Cluster IP/Port, Server IP/Port is in the request a server is selected using the Load balancing Policy/Algorithm.
Cookie 2:Cluster IP, Server IP	**Cookie 1:Cluster IP, Server IP /Port**	A server is selected based on the cookie. If no cookie or a cookie other then Cookie 2:Cluster IP, Server IP or Cookie 1:Cluster IP, Server IP /Port is in the request a server is selected using the Load balancing Policy/Algorithm.
Cookie 2:Cluster IP, Server IP	**Cookie 2:Cluster IP, Server IP**	invalid configuration

Server Side Encryption

Note - Server Side Encryption is not supported on GX Series Equalizers.

In a potentially dangerous scenario, you may be load balancing traffic and forwarding it to back-end servers along untrusted paths. Vital credit card and personally identifying information could be vulnerable during its back-end transit to clients unless you-encrypt it. Server Side Encryption (SSE) provides you with the ability to configure a cluster and/or match rule so that traffic between Equalizer and back end servers is encrypted using SSL/TLS, eliminating the untrusted paths.

A client’s HTTPS request is encrypted along its path from the client to Equalizer. Equalizer terminates the SSL/TLS connection with the client, decrypts the client request using a certificate and key and then forwards unencrypted HTTP traffic to the servers. When the server replies, the server connects with Equalizer via clear text HTTP. Equalizer, then encrypts the response and forwards it via HTTPS back to the client. Using SSE, the vulnerable path between your appliance and servers can be encrypted by enabling cluster options.

With Equalizer, Match Rules extend the Layer 7 load balancing capabilities of HTTP and HTTPS clusters by allowing you to define a set of logical conditions which, when met by the contents of the request, trigger the load balancing behavior specified in the match rule.You have the option of utilizing this intelligence as you have the capability of encrypting packets specifically identified by the match rule definitions.

The illustration below shows the SSL/TLS handshake process used with an encrypted connection between a client and Equalizer and the same process with SSE enabled between Equalizer and servers.

Equalizer provides configuration options, whereby you could encrypt all traffic between the servers and your appliance or content-specific traffic, based on a match rule.The table below explains possible Cluster/Match Rule encryption scenarios:

Cluster/Match Rule Encryption Enabled	Usage
Cluster Enabled/Match Rule Enabled	Used to encrypt all packet transfers between Equalizer and all of your servers.
Cluster Enabled/Match Rule Disabled	Used to encrypt all packet transfers from Equalizer, regardless of match rule definitions.
Cluster Disabled/Match Rule Enabled	Used to encrypt only those packets specified by the enabled match rule definition.

General Configuration Process

Note - Server Side Encryption is not supported with servers using IPv6.

The general configuration process for configuring your appliance for SSE is:

1. Set the listening port on your servers.
2. Configure Equalizer's **Global > Server Side Encryption** Cipher Suite and TLS parameters.
3. Configure cluster options for SSE.
4. Configure match rule options for SSE.

Note -The **spoof** and **TCP Multiplexing** (`tcp_mux`) options will not be available on http or https clusters or match rules if the **Server Side Encryption** (`sse`) option is enabled on the cluster and/or match rule

Proceed with the following to configure your appliance for SSE using the CLI and GUI:

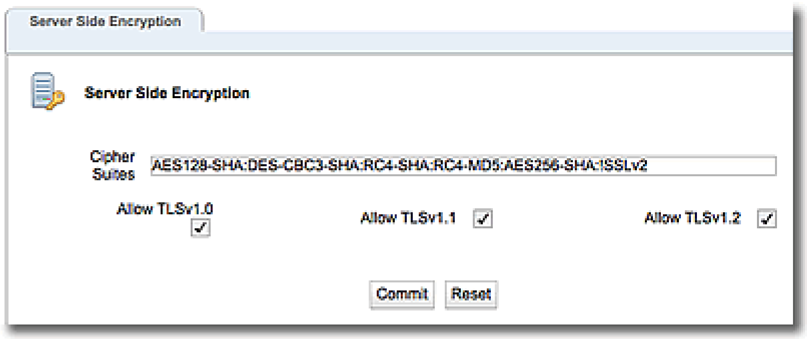
Configuring SSE Using the GUI

1. Log in to the GUI.

Global Cipher Suite and TLS Configuration

First, you will need to enable SSE on your Equalizer on a global level.

2. Select **System > Global > Server Side Encryption** on the left navigational pane.The following will be displayed on the right configuration pane.



Cipher Suites: AES128-SHA:DES-CBC3-SHA:RC4-SHA:RC4-MD5:AES256-SHA:!SSLv2
Allow TLSv1.0 ✓ Allow TLSv1.1 ✓ Allow TLSv1.2 ✓

3. Enter the cipher suite (set of cipher specifications) to use in the encryption in the **Cipher Suites** box. A default cipher suite is used by default (AES128-SHA:DES-CBC3-SHA:RC4-SHA:RC4-MD5:AES256-SHA:!SSLv2).

Note - SSLv2 is not supported as Equalizer will not negotiate with packets using SSLv2 encyrption.

4. Add additional **Cipher Suites** as described in "Layer 7 SSL Security (HTTPS Clusters)" on page 384 as necessary.

5. Enable each TLS version that you wish to use. For example, if you select only **Allow TLSv1.1**, this will be the only allowable TLS version used with the **Cipher Suite**. Select **Allow TLSv1.0** and/or **Allow TLSv1.2** as needed. At least one encryption type must be selected.

6. Click on **Commit** to save your settings.

Configuring SSE Using the CLI

Set the Server Listening Port

1. Verify that your back-end servers are configured for encrypted connections — if they are not, the connection will fail. Configure the listening port number (typically port 443 for HTTPS) for each server. Refer to "Adding and Modifying Servers" on page 529 for details.

Global Cipher Suite and TLS Configuration

First, you will need to enable SSE on your Equalizer on a global level.

2. Enter the cipher suite (set of cipher specifications) to use in the encryption.

```
eqcli > sse cipherspec cipher_spec
```

   cipher_spec is the cipher suite to use. This is passed from the client to the server in the Client Hello message. It contains the combinations of cryptographic algorithms supported by the client in order of the client's preference (first choice first). Each cipher suite defines both a key exchange algorithm and a cipher spec. The server selects a cipher suite or, if no acceptable choices are presented, returns a handshake failure alert and closes the connection.Once you add an https cluster, a default cipher suite will be added (AES128-SHA:DES-CBC3-SHA:RC4-SHA:RC4-MD5:AES256-SHA:!SSLv2).

   Note - SSLv2 is not supported as Equalizer will not negotiate with packets using SSLv2 encyrption.

   Add additional cipher specs as described in "Cluster and Match Rule Commands" on page 171 and "Layer 7 SSL Security (HTTPS Clusters)" on page 384 as necessary.

3. Now, enter the allowable TLS versions for use with the `cipher_spec`.

```
eqcli > sse flags tls_flags
```

   where **tls_flags** can be **allow_tls10** (TLS version 1.0), **allow_tls11** (TLS version 1.1) or **allow_tls12** (TLS version 1.2). You must add each TLS version that you wish to use. For example, if you add only TLS version 1.1, this will be the only allowable TLS version used with the cipher spec.

Layer 7 Cluster Reporting

Refer to "Cluster and Match Rule Statistics and Reporting (CLI and GUI)" on page 466 for details.

Layer 7 Cluster Timeouts

The Layer 7 Cluster Timeouts screen is used to configure timeouts used in cluster connection with clients and servers.

Timeouts can be configured using either the GUI or the CLI.

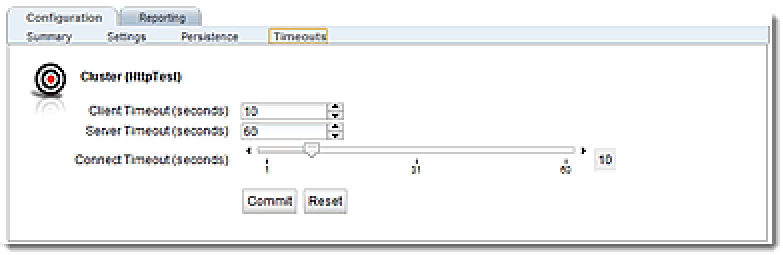
Parameters

The table below shows the timeouts used in the configuration of Layer 7 clusters.

GUI Parameter (CLI Parameter)	Description
Client Timeout (`clientto`)	The time in seconds that Equalizer waits before closing an idle client connection. The default is the global value. (between 1 and 65535 seconds)
Server Timeout (`serverto`)	The time in seconds that Equalizer waits before closing an idle server connection. The default is the global value. (between 1 and 65535 seconds)
Connect Timeout (`connto`)	The time in seconds that Equalizer waits for a server to respond to a connection request. The default is the global value.

Configuring Layer 7 Cluster Timeouts Using the GUI

Layer 7 Cluster timeouts are configured on the **Timeouts** screen. This can be accessed by selecting **Load Balance > Clusters** and then selecting a cluster. Clicking on the **Timeouts** tab will display the following.



Client Timeout (seconds) 10
Server Timeout (seconds) 60
Connect Timeout (seconds) 10

Use the table above for timeout parameters. Click on the **Commit** button after making changes to the settings.

Configuring Layer 7 Cluster Timeouts Using the CLI

Cluster timeouts can be configured in the CLI either globally or in cluster context. Enter parameters using the following format.

```
eqcli > cluster clustername {cliento|serverto|connto value
```

Use the table above for descriptions of the parameters and values.

Where:

clustername - is the the name fo the cluster.

value - is the value associated with the client timeout, server timeout, or connection timeout.

Use the table above for parameters and values .

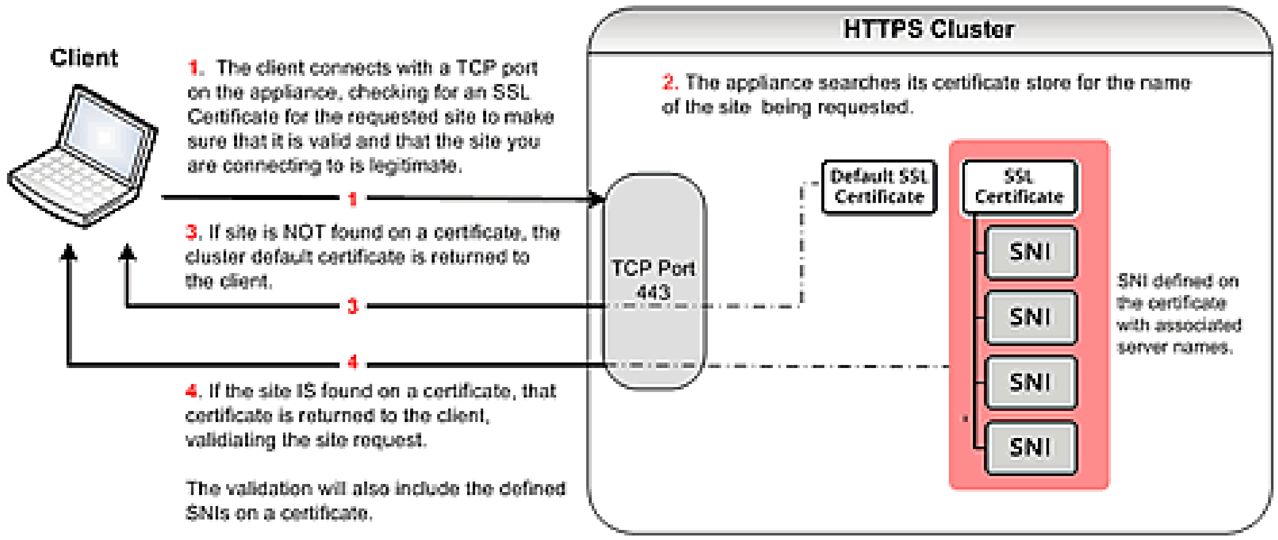
Refer to "Cluster and Match Rule Commands" on page 171 for additional information on using cluster commands in the CLI.

Server Name Indication

Server Name Indication (SNI) is an extension to the SSL and TLS protocols that indicates a server name or website that a client is attempting to connect with at the start of the handshake process. It allows a server to present multiple certificates on the same IP address and port number, thus allowing multiple secure (HTTPS) websites to be server of the same IP address while allowing all of those sites to have unique certificates all serviced on the same cluster/IP address.

SNI objects are added to certificates that are in the certificate store on Equalizer and are configured on HTTPS clusters.After a client connects with a TCP port on the load balancer, it searches it's certificate store for the website name that was exchanged as part of the HTTPS packet header. If the website is NOT presented on a certificate, the cluster's default certificate will be returned to the client. If the website IS presented on a certificated, that certificate will be returned to the client. Using SNI, additional websites are associated with certificates allowing a certificate to be returned to a client for multiple website requests, thus minimizing the need to purchase costly wild card certificates.

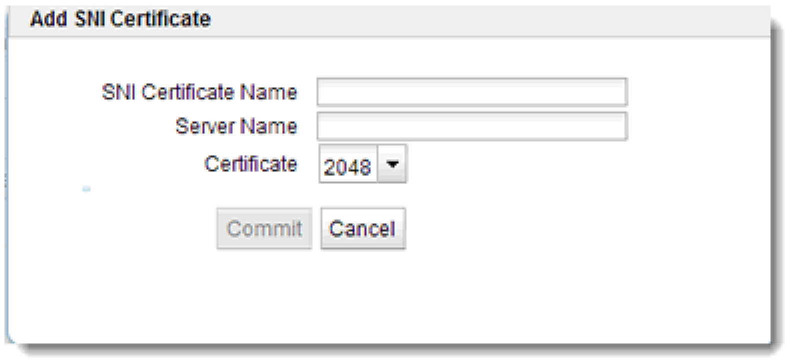
The following illustration shows the connection and certificate process with Equalizer and an HTTPS cluster:


HTTPS Cluster
Client
1. The client connects with a TCP port on the appliance, checking for an SSL Certificate for the requested site to make sure that it is valid and that the site you are connecting to is legitimate.
2. The appliance searches its certificate store for the name of the site being requested.
3. If site is NOT found on a certificate, the cluster default certificate is returned to the client.
4. If the site IS found on a certificate, that certificate is returned to the client, validiating the site request.
The validation will also include the defined SNIs on a certificate.
1
3
4
TCP Port 443
Default SSL Certificate
SSL Certificate
SNI
SNI
SNI
SNI
SNI defined on the certificate with associated server names.

Server Name Indication Using the GUI

Proceed with the following to configure SNI certificates on an HTTPS cluster using the GUI:

1. Configure an HTTPS cluster on Equalizer. Use the GUI as described in "Adding and Deleting Clusters" on page 337
2. Add a default certificate to the cluster.as described in "Layer 7 Security Certificate Screen (HTTPS Clusters)" on page 380 if one has not been added previously.
3. Upload additional certificates and their associated key files to Equalizer's file store as described in "Certificates" on page 252.
4. Select the HTTPS cluster from the left navigational pane if it is not already selected. Select the **Security SNI** tab to display the list of configured SNI as shown below. All previously configured SNI will be listed on accordion tabs.
5. To add an SNI click on ➕ to add a new SNI. The following will be displayed.



Add SNI Certificate

SNI Certificate Name	
Server Name	
Certificate	2048

6. Configure SNI parameters as follows:

Parameter	Description
SNI Certificate Name	The name of the SNI object. (up to 47 ASCII characters and can include period (.), dash (-), and underscore (_))
Server Name	The name of the website that you would like the SNI certificate to be associated with
Certificate	Use the drop down list to select the name of a certificate that you would like to associate the SNI with.

7. Click on **Commit** to save the SNI where it will be displayed on the accordion list on the **SNI** tab.
8. Add additional SNI objects to certificates as necessary.There is no maximum limit to the number of SNI objects that can be associated with each certificate. If you would like to remove an SNI, select the accordion tab on the SNI screen and click on the 🗑 button.

Server Name Indication Using the CLI

Proceed with the following to configure SNI certificates on an HTTPS cluster using the CLI:

1. Configure an HTTPS cluster on Equalizer. Use the CLI syntax described in "Cluster and Match Rule Commands" on page 171.
2. Add a default certificate to the cluster if one has not been added previously. Use the CLI syntax described in "Cluster and Match Rule Commands" on page 171.
3. Use the following CLI syntax to upload other certificates and the associated key files to Equalizer's file store.

```
eqcli > cert certname
eqcli cert-certname> certfile {edit|url}
```

   Do the same for the associated key files:

```
eqcli > cert certname
eqcli cert-certname> keyfile {edit|url}
```

4. Add an SNI object by entering the following in the HTTPS cluster context. The SNI name can be up to 47 ASCII characters and can include period (.), dash (-), and underscore (_).

```
eqcli cl-HTTPS*> sni testsni
eqcli cl-HTTPS*-sni-tes*>
```

5. Now associate certificates with the new SNI by entering the following in the SNI context:

```
eqcli cl-NEW* > sni testsni
eqcli cl-NEW*-sni-tes*> certificate snicertificate1
eqcli cl-NEW*-sni-tes*>
```

   where:

   testsni is the name of the SNI

   snicertificate1 is the name of the certificate being added to the SNI.

6. Display the contents of the new certificate by entering the following. Note that the SNI **svname** has not yet been entered.

```
eqcli cl-NEW*-sni-testsni> show
SNI Name : test
Certificate : snicertificate1
Flags :
SNI svname :
eqcli cl-NEW*-sni-test>
```

7. Add the name of the website that you would like the SNI certificate to be associated with by entering the following in the SNI context:

```
eqcli cl-NEW*-sni-testsni> sni_svname www.march22.com
eqcli cl-NEW*-sni-testsni> commit

eqcli: 12000287: Operation successful
```

8. Now verify the SNI to be sure that it is associated with a server name.

```
eqcli cl-NEW*-sni-testsni> show
SNI Name : test
Certificate : snicertificate1
Flags :
SNI svname : www.march22.com

eqcli cl-NEW*-sni-testsni>
```

9. Add additional SNI objects to certificates as necessary. There is no maximum limit to the number of SNI objects that can be associated with each certificate.

Header Editing

Header editing allows you to add, modify, and delete Layer 7 packet header data contained in client requests and server responses. You can choose to apply header editing rules on every request or response, or you can selectively apply header edits based on whether or not the client request is selected by a match rule.

Refer to "HTTP and HTTPS Header Editing Feature" on page 941 for a complete description of this feature along with examples that demonstrate how to use header editing functions in real-world scenarios.

Layer 7 TCP Cluster Settings

Layer 7 TCP clusters are used to provide IPv6 addressing for generic Layer 4 protocols, and can support IPv4 and IPv6 addressing for clusters and servers.

A key feature of EQ/OS 10 is that it is designed to allow L7 HTTP & HTTPS clusters to work with IPv6. In addition a solution is available that makes L4 clusters *also* work with IPv6. Although somewhat misleading in name, an L7 cluster is available for use with IPv6 called "L7 TCP". L7 TCP really has very little to do with HTTP or HTTPS, however, and functions as an L4 cluster.

L4 TCP clusters have special code which deals with the fact that FTP involves two TCP connections and IP addresses that are passed over the wire. The L4 code is able to rewrite those IP addresses. The L7 TCP code cannot do that, so it is NOT recommended for FTP.

The L7 TCP cluster:

- Cannot process match rules the way L7 HTTP & HTTPS clusters do.
- Cannot examine or manipulate headers
- Cannot do anything protocol-specific.

This type of cluster is essentially used to:

1. Get a TCP connection on the cluster
2. Pick a server
3. Connect the client to server

In general, the basic function of the Layer 7 TCP cluster is to provide IPv6 addressing for generic Layer 4 protocols, and support IPv4 and IPv6 addressing for clusters and servers. The functionality is very much like Layer 4 TCP cluster and should be used when IPv6 addressing is required for a TCP protocol other than HTTP or HTTPS. (The Layer 4 TCP and UDP clusters can use only IPv4 cluster addresses and can only be used with servers that have IPv4 addresses.)

For additional information on cluster types used with Equalizer refer to Cluster Types for a summary of cluster types.

The following are descriptions of the functionality and configuration parameters used with Layer 7 TCP Clusters.

Parameters

GUI Parameter (CLI Parameter)	Description
Protocol **(`proto`)**	The protocol selected in the Add Cluster form will be displayed “grayed out”.
VID **(`vid`)**	The VLAN ID number assigned to the VLAN on which the cluster resides. Refer to Common Networking Scenarios for details.

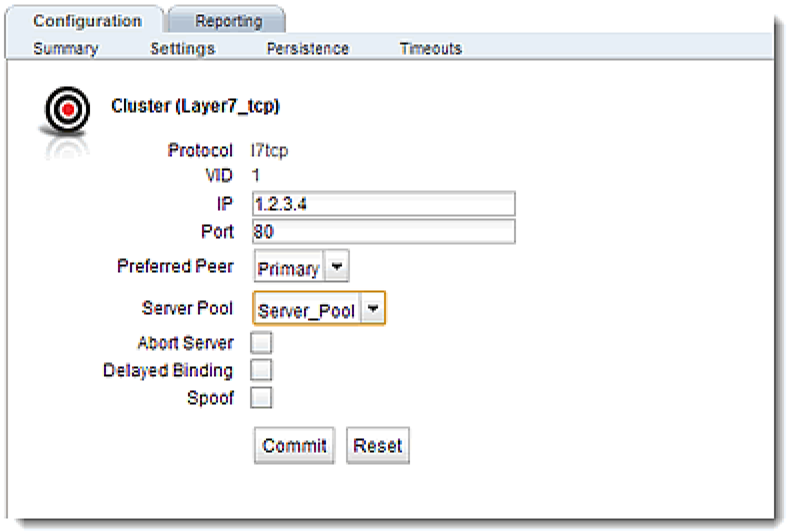
GUI Parameter (CLI Parameter)	Description
IP **(ip)**	Enter the IP address, which is the dotted decimal IP address of the cluster. The IP address of the cluster is the external address with which clients connect to the cluster.
Port **(port)**	For TCP protocol clusters the numeric port number on the Equalizer to be used for traffic between the clients and the cluster. For TCP clusters, the port defaults to 80. This port also becomes the default port for servers added to the cluster (though servers can use a different port number than the one used by the cluster).
Preferred Peer **(preferred_peer)**	Used with Active-Active Failover. Refer to Configuring Active/Active Failover Between Two Systems for details.
Server Pool **(srvpool)**	A drop down list used to select a Server Pool, or grouping of servers, to which the cluster will communicate with.
Flags	
Abort server **(abort_server)**	By default, when a client closes a connection, Equalizer waits for a response from the server before closing the server connection. If this flag is enabled, Equalizer will not wait for a response before closing the connection to the server; instead it sends a TCP RST (reset) to the server when the client closes the connection. This option will typically reduce the number of server connections in the TIME_WAIT state, as shown by the netstat console command.
Delayed Binding **(delayed_binding)**	If this option is selected, the client must send the first byte of data on newly established connection.
Spoof **(spoof)**	When the spoof option is enabled on a cluster, Equalizer uses the client's IP address as the source IP address in all packets sent to a server in that cluster. This option is enabled by default. When spoof is enabled, all server responses to client requests that came through the Equalizer cluster IP address must be routed by the server back to the client through Equalizer. In many cases, the easiest way to do this is to set the default gateway on the server with a server instance in a server pool to Equalizer's IP address on the server VLAN. If this is not possible, you can establish static routes on the server to send responses to specific client IP addresses to Equalizer's IP address on the VLAN. If you disable spoof, the server receiving the request will see Equalizer's IP address as the client address because the TCP connection to the client is terminated when the request is routed. The server will therefore send its response back to Equalizers IP address. Disabling the spoof option enables Source Network Address Translation (SNAT).

GUI Parameter (CLI Parameter)	Description
Protocol **(proto)**	The protocol selected in the Add Cluster form will be displayed "grayed out".

GUI Parameter (CLI Parameter)	Description
VID (`vid`)	The VLAN ID number assigned to the VLAN on which the cluster resides. Refer to Common Networking Scenarios for details.
IP (`ip`)	Enter the IP address, which is the dotted decimal IP address of the cluster. The IP address of the cluster is the external address with which clients connect to the cluster.
Port (`port`)	For TCP protocol clusters the numeric port number on the Equalizer to be used for traffic between the clients and the cluster. For TCP clusters, the port defaults to 80. This port also becomes the default port for servers added to the cluster (though servers can use a different port number than the one used by the cluster).
Preferred Peer (`preferred_peer`)	Used with Active-Active Failover. Refer to Configuring Active/Active Failover Between Two Systems for details.
Server Pool (`srvpool`)	A drop down list used to select a Server Pool, or grouping of servers, to which the cluster will communicate with.
Flags	
Abort server (`abort_server`)	By default, when a client closes a connection, Equalizer waits for a response from the server before closing the server connection. If this flag is enabled, Equalizer will not wait for a response before closing the connection to the server; instead it sends a TCP RST (reset) to the server when the client closes the connection. This option will typically reduce the number of server connections in the TIME_WAIT state, as shown by the netstat console command.
Delayed Binding (`delayed_binding`)	If this option is selected, the client must send the first byte of data on newly established connection.
Spoof (`spoof`)	When the spoof option is enabled on a cluster, Equalizer uses the client's IP address as the source IP address in all packets sent to a server in that cluster. This option is enabled by default. When spoof is enabled, all server responses to client requests that came through the Equalizer cluster IP address must be routed by the server back to the client through Equalizer. In many cases, the easiest way to do this is to set the default gateway on the server with a server instance in a server pool to Equalizer's IP address on the server VLAN. If this is not possible, you can establish static routes on the server to send responses to specific client IP addresses to Equalizer's IP address on the VLAN. If you disable spoof, the server receiving the request will see Equalizer's IP address as the client address because the TCP connection to the client is terminated when the request is routed. The server will therefore send its response back to Equalizers IP address. Disabling the spoof option enables Source Network Address Translation (SNAT).

Configuring a Layer 7 TCP Cluster Using the GUI

The Layer 7 TCP Cluster configuration screen can be accessed by adding a new Layer 7 TCP cluster or by selecting **Load Balance > Clusters** and then selecting the **Settings** tab on the right configuration pane. The figure below shows a Layer 7 TCP**Configuration Settings** screen.



Cluster (Layer7_tcp)

Protocol	l7tcp
VID	1
IP	1.2.3.4
Port	80
Preferred Peer	Primary
Server Pool	Server_Pool
Abort Server	☐
Delayed Binding	☐
Spoof	☐

Use the table above for parameters, values, and flags for the configuration of a Layer 7 TCP cluster. Click on the **Commit** button after making changes to the settings.

Configuring a Layer 7 TCP Cluster Using the CLI

Layer 7 HTTP and HTTPS Clusters can be configured in the CLI either globally or in cluster context. Enter parameters using the following format.

Layer 7 HTTP and HTTPS Clusters can be configured in the CLI either globally or in cluster context. Enter parameters using the following format.

```
eqcli > cluster clustername parameter value flags flag
```

Use the table above for descriptions of the parameters and values.

Where:

clustername - is the the name fo the cluster.

parameter - is the parameter.

value - is the value associated with the parameter.

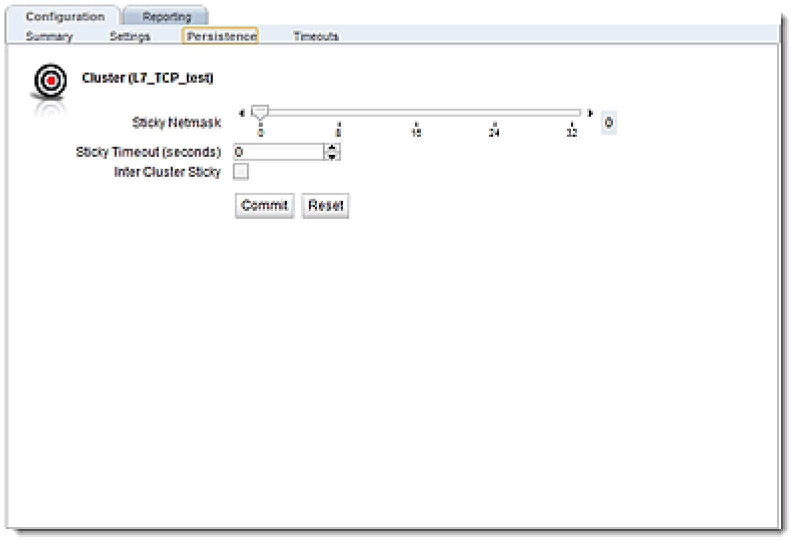
flag - is the flag to be associated with the cluster.

Use the table above for parameters and values .

Refer to "Cluster and Match Rule Commands" on page 171 for additional information on using cluster commands in the CLI.

Layer 7 TCP Cluster Persistence

Layer 7 TCP cluster persistence is the same as Layer 4 TCP cluster persistence. Refer to "TCP Cluster Persistence" on page 345 for details.



Cluster (L7_TCP_test)

Sticky Netmask 0

Sticky Timeout (seconds) 0

Inter Cluster Sticky

Additional Cluster Configuration

The Related Topics describe additional cluster configuration.

About Passive FTP Translation

In EQ/OS 8.6 if your servers were on a network that the outside world could not reach, you were provided the capability of enabling a passive FTP translation option. This option caused Equalizer to rewrite outgoing FTP PASV control messages from the servers so they could contain the IP address of the virtual cluster rather than that of the server. This was a global option.

In previous releases, an FTP cluster was required to have the "Spoof" option enabled, which means that the ADC would use the client's IP address as the source IP address in all packets sent to the servers in the cluster. Spoof can now disabled on an FTP cluster that uses passive FTP connections to servers – which means that the ADC's subnet IP address will be used as the source IP in all packets sent to servers.

Contact customer support for instructions if you require this feature to be turned OFF.

Enabling Cookies for Persistent Connections

For Layer 7 HTTP and HTTPS clusters, you can enable the **persist** check box to use cookies to maintain a persistent session between a client and a particular server for the duration of the session.

When you use cookie-based persistence, Equalizer inserts a cookie into the server's response header on its way back to the client. This cookie uniquely identifies the server to which the client was connected and is included automatically in subsequent requests from the client to the same cluster. Equalizer can use the information in the cookie to route the requests to the same server. If the server is unavailable, Equalizer automatically selects a different server.

This option is enabled by default. Also see the descriptions of the **always**, **cookie age**, **cookie domain**, and **cookie path** cluster parameters under "Modifying a Layer 7 HTTP or HTTPS Cluster" on page 362.

Enabling Persistent Server Connections

Equalizer provides several methods by which connections between clients and servers can be made persistent; that is, it is possible to route a series of requests from a particular client to the same server, rather than have the Equalizer load balance each request in the series -- potentially sending each request to a different server.

- For Layer 4 clusters, persistent server connections are enabled using the Sticky Time cluster parameter and (optionally) the Inter-Cluster Sticky cluster flag.
- For Layer 7 clusters, persistent server connections are enabled using the **Persist** and **Always** cluster flags.

Enabling Sticky Connections

For Layer 4 TCP and UDP clusters, you can use IP-address based sticky connections to maintain persistent sessions.

The **sticky time** period is the length of time over which Equalizer ensures that it directs new connections from a particular client to the same server. The timer for the sticky time period begins to expire as soon as there are no active connections between the client and the cluster. If Equalizer establishes a new connection to the cluster, Equalizer resets the timer for the sticky time period.

Sticky connections are managed on Equalizer using sticky records that record the IP address, port and other information for the client-server connection. When you enable sticky connections, the memory and CPU overhead for a connection increase. This overhead increases as the sticky time period increases.

Consequently, you should use the shortest reasonable period for your application and avoid enabling sticky connections for applications unless they need it. For most clusters, a reasonable value for the sticky time period is 600 seconds (that is, 10 minutes). If your site is extremely busy, consider using a shorter sticky time period.

With the **inter-cluster sticky** option, you can configure Equalizer to direct requests from a client to the same server on any available port that has a current persistent connection *in any cluster*.

When Equalizer receives a client request for a Layer 4 cluster with inter-cluster sticky enabled and the client does not have a sticky record for the cluster, then Equalizer will check other clusters that have inter-cluster sticky enabled for a sticky record for the same client and server -- but on a different server port than the one originally used in the client request.

If such a sticky record is found and the server IP/port in the sticky record is configured as a server in the current cluster, then the sticky record is used to send the client request to that server IP/port. Otherwise, the client request is load balanced across the server pool in the cluster.

In order for the inter-cluster sticky option to work:

- The two clusters must have the same cluster IP address and different ports.
- At least one server in each of the two clusters must be configured with the same IP address and different ports.

Inter-cluster stickiness is provided for the case where you have similar services running on the same server IP on two or more ports. Using *port ranges* for a cluster achieves essentially the same effect, without using another cluster IP address (See "TCP Cluster Configuration Settings" on page 341). Using **inter-cluster sticky** is preferable in situations where you'd like the service available on multiple cluster IPs as well as multiple ports.

To enable sticky connections for a cluster, follow these steps:

1. Log into the GUI using a login that has add/del access for the cluster (See "Logging In" on page 238)
2. In the left frame, click the name of the Layer 4 TCP or UDP cluster to be configured. The cluster's parameters appear in the right frame.
3. Select the **Persistence** tab in the right frame.
4. In the **sticky time** field, specify the sticky time period in seconds greater than zero.

5. To direct all requests from a particular client to the same server even if the connection is to a different virtual cluster, check the **inter-cluster sticky** checkbox. You can turn on inter-cluster stickiness only if you have enabled sticky connections by specifying a **sticky time** greater than zero.
6. Click the **commit** button.

Enabling the Once Only and Persist Options

Since HTTP 1.0, web browsers and servers have been able to negotiate persistent connections over which multiple HTTP transactions could take place. This is useful when several TCP connections are required in order to satisfy a single client request.

For example, before HTTP 1.1, if a browser wished to retrieve the file *index.html* from the server www.coyotepoint.com, the browser would take the following actions:

1. Browser opens TCP connection to www.coyotepoint.com.
2. Browser sends request to server **"GET /index.html".**
3. Server responds with the content of the page (HTML).
4. Server closes connection.
5. Browser determines that there are objects (images) in the HTML document that need to be retrieved, so the browser repeats Steps 1 to 4 for each of the objects.

There is a lot of overhead associated with opening and closing the TCP connections for each image. The way HTTP 1.0 optimizes this is to allow multiple objects (pages, images, etc) to be fetched and returned across one TCP socket connection. The client requests that the server keep the connection open by adding the request header **Connection: keep-alive** to the request. If the server agrees, the server will also include **Connection: keep-alive** in its response headers, and the client is able to send the next request over the persistent HTTP connection without the bother of opening additional connections.

For HTTP/1.1, persistent connections are the default.

For a Layer 7 cluster, Equalizer evaluates (and possibly changes) both the request and response headers that flow between the client and server (the request and response bodies are not examined). Match rules are applied to each client header, cookies may be inserted, and headers may be rewritten. When a client includes **keep-alive** in its headers, there is a fair amount of work required by the Equalizer to determine when the next set of request headers is ready to be parsed (evaluated), since there may be quite a lot of data going across the connection between sets of headers.

To reduce this workload, the **once only** flag instructs the Equalizer to evaluate (and potentially modify) only the *first* set of headers in a connection. So, in our example above, only the headers in the request for the *index.html* file are evaluated; the subsequent requests to obtain the images are not load balanced, but sent to the same server as the first request.

Enabling **once only** can be incompatible with persistence and Layer 7 HTTPS clusters (which rewrite HTTP to HTTPS links in server response headers), since in these cases we generally want to examine every request in a connection. However, in configurations where examining the headers in every transaction in a connection is not required, enabling **once only** can significantly improve performance.

Whether **once only** is enabled or not has a significant effect on how Equalizer routes requests, as summarized in the following table:

Requests in a single keep-alive connection	**once only enabled**	**once only disabled**
First Request		
persist enabled	If request contains a cookie and there is no match rule hit, send request to the server in the cookie. If request contains a cookie and there is a match rule hit, send the request to the server in the cookie *only if it is in the list of servers selected in the match rule definition*. Otherwise, ignore the cookie. If there is no cookie, load balance the request and send to the server chosen.	If request contains a cookie and there is no match rule hit, send request to the server in the cookie. If request contains a cookie and there is a match rule hit, send the request to the server in the cookie *only if it is in the list of servers selected in the match rule definition*. Otherwise, ignore the cookie. If there is no cookie, load balance the request and send to the server chosen.
persist disabled	Load balance the request and send to the server chosen.	Load balance the request and send to the server chosen.
match rule hit	Send to the server chosen by the match rule.	Send to the server chosen by the match rule.
Subsequent Requests		
persist enabled	Send to same server as *first* request (any cookie in request is ignored).	If request contains a cookie, send request to the server in the cookie. If there is no cookie, load balance request and send to server chosen by policy.
persist disabled	Send to same server as *first* request.	Load balance the request and send to the server chosen.
match rule hit	Send to same server as *first* request.	Send to the server chosen by the match rule.

For example, let’s look at how Equalizer processes HTTPS requests. For an HTTPS cluster, Equalizer off loads SSL processing from the server pool in the cluster; that is, Equalizer does all the SSL related processing itself, and then forwards the request in HTTP to the server. When it does this, it inserts special headers into the request to indicate that the request was received by Equalizer in HTTPS and processed into HTTP (see "HTTPS Header Injection" on page 422). If **once only** is set, these special headers are only inserted into the *first* request in a connection; the remainder of the requests in the connection are still processed, but no headers are inserted. Most servers that support SSL off loading require that every request contain the special headers -- therefore, in most cases like this you need to disable the **once only** flag for the cluster if you want to be able to parse for these headers in every request on the server end.

The **once only** flag is enabled by default when adding an L7 cluster. In general, it is more efficient to enable **once only**; but, in situations where load balancing decisions need to be made for every request or where any of the above effects are undesirable, **once only** should be disabled.

Note - Although it is permitted by the software, it is *not* recommended to define a Layer 7 cluster with **persist** and **once only** both turned off, and with no match rules. By defining a Layer 7 cluster in such a way, you are essentially disabling Layer 7 processing, while still incurring extra overhead for the Layer 7 cluster. If your application requires a cluster with no persistence, header processing, or match rules, then we recommend that you define a Layer 4 UDP or TCP cluster for the best performance.

Enabling Both the Once Only and Always Options

The **always** flag influences when Equalizer inserts cookies into server responses; it in turn is affected by the setting of the **once only** flag, as shown in the following table:

	once only enabled	**once only disabled**
always enabled	Equalizer always inserts a cookie into the first set of response headers on a connection *only*. The cookie is inserted regardless of whether the server included one in the response. Subsequent responses on the same connection are forwarded to the client *unchanged* by Equalizer.	Equalizer inserts its own cookie into *all* server responses on a connection. The cookie is inserted regardless of whether the server included one in the response.
always disabled	If the *first* server response on a connection already has a server cookie in it, Equalizer inserts its own cookie into the *first* set of response headers on the connection. If the response has no cookie in it, Equalizer does *not* insert one of its own. Subsequent responses on the same connection are forwarded to the client *unchanged* by Equalizer.	If the *first* server response on a connection already has a server cookie in it, Equalizer inserts its own cookie into the *first* set of response headers on the connection. Equalizer will insert a cookie into subsequent responses on the same connection if: they do not contain a valid cookie the **cookie generation** has changed the server in the cookie has the **quiesce** flag enabled

Note - the cluster parameters cookie path, cookie age, cookie generation, and cookie domain specify cookie content for the cluster. If any of these parameters are updated, this changes the information used in the cookies that Equalizer inserts into server responses.

Enabling Once Only and Compression

Enabling both the **once only** and **compress** options is not allowed by the GUI. These two options are not compatible, since setting them both would mean that only the first response in a connection would be compressed and not the remainder of the responses, which would likely cause client errors.

Enabling Once Only and No Header Rewrite for HTTPS

In a Layer 7 HTTPS cluster, clients connect to the cluster IP using HTTPS connections. Equalizer terminates the HTTPS connection and communicates with the server pool in the cluster using the HTTP protocol. By default, Equalizer examines server responses for `http://`URLs and rewrites them as `https://` URLs, so that these URLs work properly on the client. If, for example, a server sends an HTTP redirect using the `Location:` header, this URL most likely will include the `http://` protocol. Equalizer rewrites this response so that the URL uses `https://`.

For server connections that contain multiple server responses, the setting of the **once only** flag determines whether Location: headers in all server responses are rewritten. This is shown in the table below.

Note that the GUI does not permit you to enable **once only** and disable **no header rewrite** -- this option combination would rewrite the `Location:` header in only the first response in the connection, and not rewrite the headers in subsequent responses in the same connection. Doing so would produce errors on the client.

Of course, you can also direct Equalizer to pass responses from the server without rewriting URLs by enabling the **no header rewrite** flag on the cluster.

	once only enabled	**once only disabled**
no header rewrite disabled	Not supported.	The Location: headers of *every* response in a connection are rewritten.
no header rewrite enabled	No headers are rewritten.	No headers are rewritten.

Specifying a Custom Header for HTTP/HTTPS Clusters

Some applications require specific headers in incoming client requests, and Equalizer provides the custom header field in HTTP and HTTPS clusters to allow you to inject a custom header into the client request before it is sent to a server behind Equalizer.

An example is the Exchange 2003 version of Microsoft Outlook Web Access (OWA). OWA 2003 normally requires that all incoming client requests use the Secure Sockets Layer (SSL) protocol. This means that all client requests must have the `https://` protocol in the URI. If, however, OWA is running on a server in an Equalizer Layer 7 HTTPS cluster, then OWA will receive all requests with `http://` in the URI, since Equalizer performs SSL processing before passing the requests on to the server.

OWA 2003 allows for SSL off loading through the use of a special header, as explained in the following Microsoft technical article:

```
http://technet.microsoft.com/en-us/library/578a8973-dc2f-4fff-83c6-
39b1d771514c.aspx
```

Two things are necessary when running OWA 2003 behind Equalizer:

1. Configure OWA to watch HTTP traffic for requests containing a custom header that indicates that the request was originally an SSL request that was processed by SSL off loading hardware (i.e., Equalizer) before reaching OWA (see the above article for instructions)
2. Configure the Equalizer cluster to add the custom header to all requests before sending them on to the OWA server (this is explained below)

The following procedure shows you how to add a custom header to an existing HTTPS cluster definition, using the header required for an OWA 2003 server as an example.

3. Log into the GUI using a login that has add/del access for the cluster (See "Logging In" on page 238.)

4. In the left frame, click the name of the cluster to be configured.
5. In the right frame, select the **Configuration Required** tab.
6. Type the following in the **custom header** field:

```
Front-End-Https: on
```

7. Select **commit** to modify the cluster.

Performance Considerations for HTTPS Clusters

Layer 7 HTTPS clusters have several options that can have a significant impact on the performance and behavior of the cluster:

1. The injection of a **customheader** to provide transaction-specific information to the server. For example, to tell the server that Equalizer terminated the HTTPS connection and performed SSL processing on the incoming request (see the previous section, above).
2. The "munging", or translation, of HTTP redirects to HTTPS redirects (see the description of the **no header rewrite** flag under Modifying a Layer 7 Virtual Cluster).
3. The **once only** flag. This flag is present to speed up processing of HTTP requests by only looking at the first request, but since HTTPS has a lot of overhead associated with it anyway, turning this flag off does not reduce HTTPS performance. Furthermore, having this flag on for HTTPS clusters causes some applications to not function as needed.

In general, it is recommended to turn the **once only** flag off for HTTPS clusters. In order to inject custom headers and rewrite headers in every transaction in a connection, turning off **once only** is required.

HTTPS Performance and Xcel SSL Acceleration

The E650GX and E450GX include the Xcel SSL Accelerator Card. Equalizer models without Xcel (E250GX and E350GX) performs all SSL processing in software using the system CPU. Equalizers with Xcel perform all SSL processing using the dedicated processor on the Xcel card. This allows the system CPU to concentrate on non-SSL traffic. For most applications, Xcel will process several hundred HTTPS transactions per second with no noticeable degradation in performance either for the HTTPS cluster or for Equalizer as a whole.

In terms of bulk data throughput, the theoretical maximum throughput for Xcel/HTTPS is roughly 50% of that for the Equalizer in HTTP mode: Equalizer models with gigabit Ethernet can move HTTP traffic at wire speed (1Gbit/s) for large transfers, while Xcel can encrypt only approximately 400Mbit/s with 3DES/SHA1 or 600Mbit/s with RC4/MD5. This reflects the fact that Xcel is primarily a transaction accelerator, not a bulk data encryption device. It is noteworthy, however, that even when moving bulk data at 600Mbit/s, Xcel removes the entire load of HTTPS/SSL processing from the server pool in the cluster.

One final issue to be aware of is that Xcel supports only 3DES and RC4 encryption; it does not support AES. It also does not support SSL or TLS cipher suites that use ephemeral or anonymous Diffie-Hellman exchange (cipher suites whose names contain "EDH", "DHE", or "ADH").

The default configuration for HTTPS clusters created with Xcel enabled will not use the modes described above. If, however, one either modifies the cluster’s cipher suite string to use them, it is possible that they may be negotiated with clients. This will not lead to incorrect operation of the system, but encryption for these cipher suites will occur in software instead of taking advantage of the improved performance provided by the Xcel hardware.

HTTPS Header Injection

When a connection is established by a client for an HTTPS cluster, Equalizer performs the SSL processing on the request (this is called SSL off loading), and adds some additional headers to the client's request before forwarding the request on to a server:

X-LoadBalancer: Equalizer

X-Forwarded-For: (client's IP address)

If the client provides an SSL certificate, the following are also added:

X-SSL-Subject: (certificate's X509 subject)

X-SSL-Issuer: (certificate's X509 issuer)

X-SSL-notBefore: (certificate not valid before info)

X-SSL-notAfter: (certificate not valid after info)

X-SSL-serial: (certs serial number)

X-SSL-cipher: (cipher spec)

If these headers are present in a request received by a server, then the server knows that the request was originally an HTTPS request and was processed by Equalizer before being forwarded to the server.

These headers are inserted into every request if the **once only** flag is disabled; if **once only** is enabled, then only the first request in a connection will have these headers inserted.

Some application may require a special header in the request, and the following section describes how Equalizer can be configured to provide a custom HTTPS header for such applications.

Providing FTP Services on a Virtual Cluster

The FTP protocol dates from the 1970s, and was designed to be used in an environment where:

- the network topology is simple
- the FTP server and client communicate directly with one another
- the addresses used by the client and server for active FTP data connections can be negotiated over the FTP control connection
- the FTP server is able to make connections back to the FTP client

These operational characteristics of FTP require special configuration for load balancers (as well as firewalls and NAT devices) that pass traffic between FTP servers and FTP clients:

- NAT devices and routers (including load balancers like Equalizer) on the client and server sides must be configured to monitor FTP transactions and provide appropriate address translation and packet rewriting.
- Firewalls on the client and server sides must be configured to let traffic on the ports used for FTP through the firewall.

Consult the documentation for the firewalls and NAT devices used at your site to determine how to set up those devices appropriately for FTP transfers. See the next section for how to configure an Equalizer cluster for responding to FTP requests from clients.

FTP Cluster Configuration

When configuring an FTP cluster on Equalizer, the following guidelines must be followed:

- The **protocol** for the cluster must be **Layer 4 TCP**.
- The **start port** parameter for the cluster must be set to port **21**. (Note that port 20 is also used, but you do not specify it when adding the cluster.)
- The **spoof** flag must be set according to whether FTP clients connecting to the cluster will be using ACTIVE or PASSIVE mode:
  - If all clients will be connecting using PASSIVE mode, then the spoof flag can be either enabled or disabled as desired.
  - If some or all clients will be connecting using ACTIVE mode, then the **spoof** flag must be enabled. ACTIVE mode will not work if the **spoof** flag is disabled.

FTP data connections are automatically configured (internally) with a **sticky time** of one second. This is necessary to support the passive mode FTP data connection that most web browsers use. This means that there will be one sticky record kept for each FTP data connection. For an explanation of sticky records, see "Enabling Sticky Connections" on page 413"Enabling Sticky Connections" on page 413

- FTP clusters occupy two internal virtual cluster slots, even though only one appears in the interface. This permits Equalizer's NAT subsystem to rewrite server-originated FTP data connections as they are forwarded to the external network.
- You cannot enable the **direct server return** option on an FTP cluster.

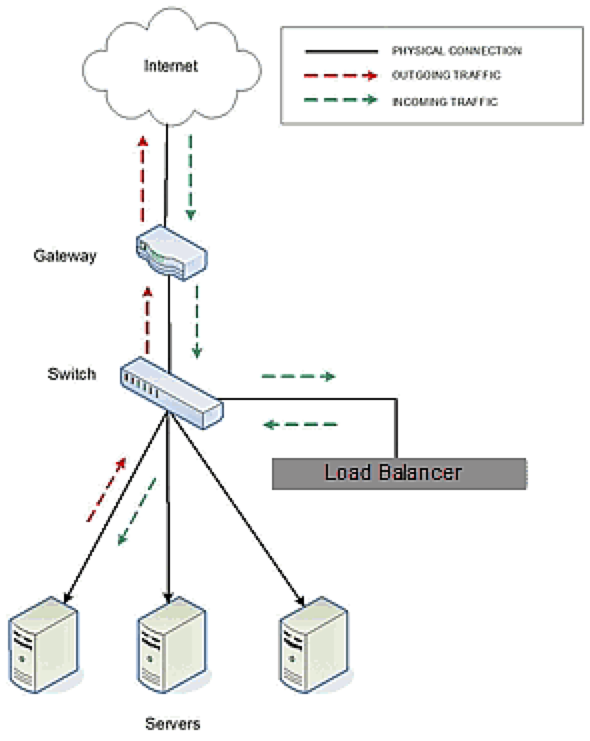
Configuring Direct Server Return

In a typical load balancing scenario, server responses to client requests are routed through Equalizer on their way back to the client. Equalizer examines the headers of each response and may insert a cookie, before sending the server response on to the client.

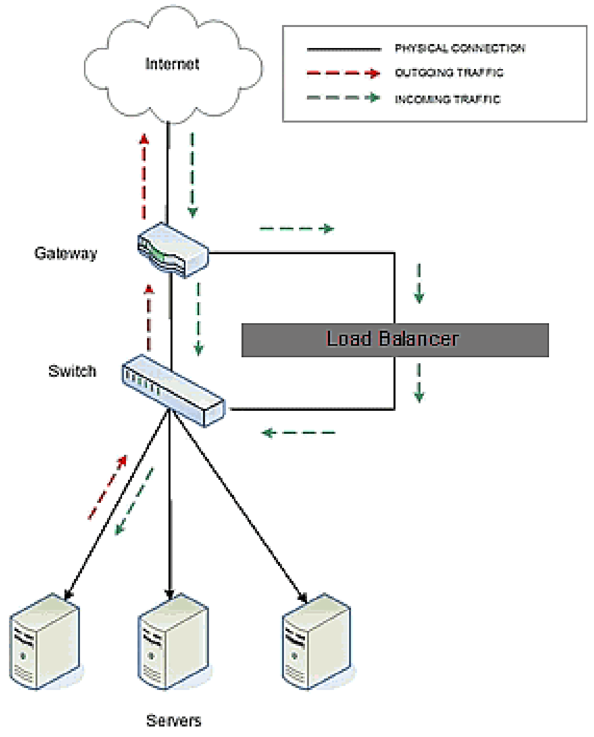
In a Direct Server Return (DSR) configuration, the server receiving a client request responds directly to the client IP, bypassing Equalizer. Because Equalizer only processes incoming requests, cluster performance is dramatically improved when using DSR in high bandwidth applications, especially those that deliver a significant amount of streaming content. In such applications, it is not necessary for Equalizer to receive and examine the server's responses: the client makes a request and the server simply streams a large amount of data to the client.

DSR is supported on Layer 4 TCP and UDP clusters only, and is not supported for FTP clusters (Layer 4 TCP clusters with a start port of 21). Port translation or port mapping is not supported in DSR configurations.

DSR configurations are usually configured in single network mode, where the cluster IP and the server IPs are all on the internal interface. An example single network mode DSR configuration is shown below:



DSR can also be used in dual network mode, although this is a less common configuration than single network mode. Cluster IPs are on the external interface, and server IPs are on the internal interface. An example of a dual network mode DSR configuration is shown below.



Note - In both configurations that the incoming client traffic is assumed to originate on the other side of the gateway device for the subnets on which Equalizer and the servers reside. The servers will usually have their default gateway set to something other than Equalizer so that they can respond directly to client requests.

The cluster parameters **Direct Server Return, Spoof**, and **Idle Timeout** are directly related to direct server return connections:

- **Direct Server Return -** this option enables Direct Server Return. All requests to this cluster IP will be forwarded to the server with the client IP as the source IP, and the cluster IP as the destination IP. The loopback interface of the server must be configured with the cluster IP to receive the requests. See "Configuring Servers for Direct Server Return" on page 183.
- **Spoof -** this option causes Equalizer to spoof the client IP address when Equalizer routes a request to a server in a virtual cluster; that is, the IP address of the client is sent to the server, not the IP address of the Equalizer. This flag must be enabled for DSR.
- **Idle Timeout -** The is the time in seconds before reclaiming idle Layer 4 connection records. Applies to Layer 4 TCP clusters only. For DSR the **Idle timeout** slider must be set to a non-zero value, or Equalizer will never reclaim connection records for connections terminated by the server. The cluster's **Idle Timeout** should be set to the longest period within your application that you would like Equalizer to wait for consecutive messages from the client (since the Equalizer does not see server packets on DSR connections). For example, if the longest expected server response time and the longest expected delay between client responses on active connections are both 60 seconds, then set the **Idle Timeout** slider to 120 seconds.

To create a new cluster or modify an existing one for DSR, do the following:

1. Log into the GUI using a login that has add/del access for the cluster (See "Logging In" on page 238.)
2. Do ***one*** of the following:
    a. Create a new Layer 4 TCP or UDP cluster: right-click **Equalizer** in the left navigational pane and select **Add Cluster.** After you enter and **commit** the basic information, you'll be taken to the server **Configuration** tab.
    b. Modify an existing Layer 4 TCP or UDP cluster: click on the cluster name in the left frame to display the cluster's **Configuration** tab in the right frame.
3. Enable the **Direct Server Return** and **Spoof** check boxes.
4. If the cluster is a Layer 4 TCP cluster and the **idle timeout** parameter is set to **0**, increase it as described in the table above. Skip this step for Layer 4 UDP clusters.
5. Click on **Commit** to save your changes to the cluster configuration.
6. If you need to add server instances to a server pool, add them by doing the following:
    a. Right-click the server pool name in the left navigational pane frame and select **Add Server Pool**.
    b. Fill in the remainder of the required information.
    c. Click on the **Commit** button to save your entries.
7. Perform the procedure in the following section on each server that you add to the cluster.

Testing Your Basic Configuration

Once you have installed and configured Equalizer and your servers, perform tests to verify that Equalizer is working properly.

To perform these tests, you need the following:

- A test machine on the internal network (the same physical network as the servers; one of the server machines can be used for this purpose).
- If you have a two-network configuration, a test machine on the external network.
- A client machine somewhere on the Internet, to simulate a “real-world” client. This machine should be set up so that the only way it can communicate with your servers or Equalizer is through your Internet router.

Then follow these steps:

1. Ping Equalizer’s external address (if configured) from a host on the external network interface address.
2. Ping Equalizer’s internal address from a host on the internal network interface address.
3. If DNS is configured, ping a host on the Internet (e.g., `www.coyotepoint.com`) from Equalizerto ensure that DNS and the network gateway are functioning properly.
4. From the internal-network test machine, ping the physical IP address of each server. You should be able to successfully ping all of the servers from the test machine.
5. From the internal-network test machine, ping the server aliases on each of the servers. You should be able to successfully ping all of the servers from the test machine using their aliases.
6. From the internal test machine and each of the servers, ping the Equalizer address that you use as the default gateway on your servers. (If you use a two-network topology, this will be Equalizer’s internal address or failover alias.)
7. From the internal-network test machine, connect to the server aliases on service ports of running daemons (you may need to configure telnet or ssh services on Windows servers). You should be able to connect successfully to the server aliases.
8. If you use a two-network configuration: From the external-network test machine, ping a physical server IP address using `ping -R` to trace the route of the ping. The EqualizerIP address should appear in the list of interfaces that the ping packet traverses. You can also use the `traceroute` (UNIX) or `tracert` (Windows) tools to perform this test.
9. Log into the GUI on either the external (if configured) or internal interfaces, as described in "Logging In" on page 238.

Using Match Rules

Note - Match Rules are not supported on E250GX model Equalizers.

The ability to make load balancing decisions based on the content of a client request is what separates Layer 7 processing from the processing options available at Layer 4. For Layer 7 HTTP and HTTPS clusters, Match Rules provide fine-grained control over load balancing decisions based on the content of the client request. If you need to be able to route requests to the servers in a cluster based on the content of the request, Match Rules are the answer.

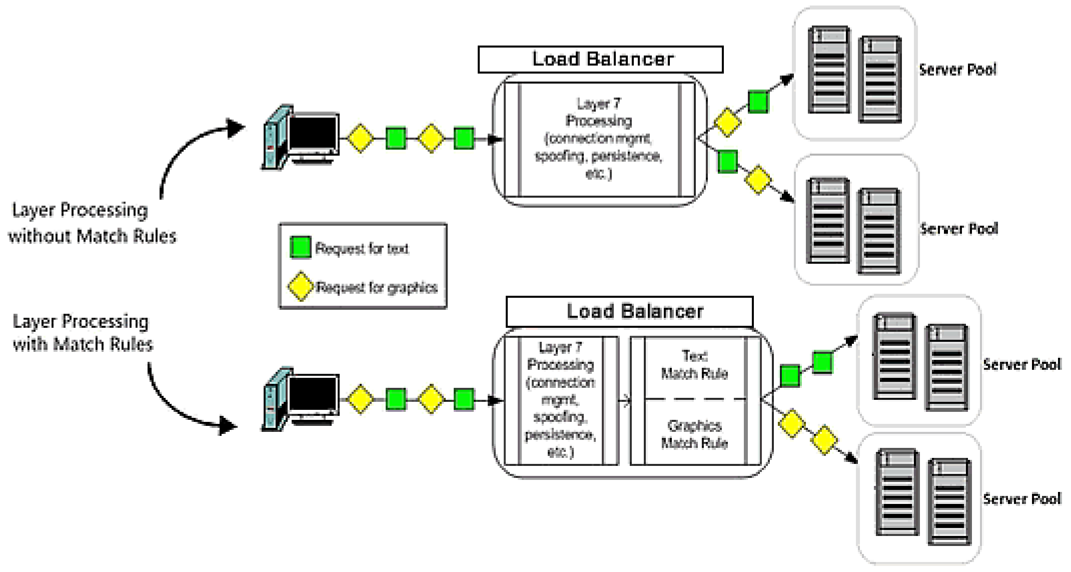
Layer 7 HTTP and HTTPS clusters can use logical constructs called “Match Rules” to control the processing of the incoming data stream from clients. Match rules extend the Layer 7 load balancing capabilities of HTTP and HTTPS clusters by allowing you to define a set of logical conditions which, when met by the contents of the request, trigger the load balancing behavior specified in the match rule.

Typically, a match rule selects the subset of servers that the load balancing algorithms will use for a particular request. By default, a request is load balanced over all the available non-spare servers in a cluster. Match rules allow you to select the group of servers, or server pools, that will be used to load balance the request.

For each virtual cluster, you can specify any number of match rules. For each match rule, you specify the subset of servers or server pools that can handle requests that meet the rule criteria.

A match rule provides custom processing of requests within connections. Equalizer provides common and protocol-specific match functions that enable dynamic matching based on a request’s contents. Protocol-specific match functions typically test for the presence of particular attributes in the current request.

For example, a Layer 7 HTTP virtual cluster can specify matching on specific path name attributes to direct requests to subsets of servers or server pools so that all requests for images are sent to the image servers. The difference between load balancing with and without match rules in such a situation is illustrated in the following figure.


Load Balancer
Layer 7 Processing (connection mgmt, spoofing, persistence, etc.)
Server Pool
Server Pool
Layer Processing without Match Rules
Request for text
Request for graphics
Layer Processing with Match Rules
Load Balancer
Layer 7 Processing (connection mgmt, spoofing, persistence, etc.)
Text Match Rule
Graphics Match Rule
Server Pool
Server Pool

Most client requests are a mix of requests for text and graphics. Layer 7 processing without Match Rules balances requests across the specified server pool so that each server instance in the server pool will see a mix of text and graphics requests. This means that all text and graphics must be available on each server pool.

Some sites may want to have one system serve only requests for graphics, and one system serve only text requests.

By adding appropriate Match Rules, Equalizer can examine each request to determine if the content requested is Text or Graphics, and send the request to the appropriate server pool. In this example, the servers need only hold the content they are serving, text or graphics.

How Match Rules are Processed

A match *rule* is like an if-then statement: an expression is evaluated and if it evaluates to true the body of the match rule applies to the request.

A match *expression* is a combination of match functions with logical operators, and can be arbitrarily complex. This allows for matching requests that have, for example:

(attribute A) AND NOT (attribute B)

If a match expression evaluates to *true*, then the data in the request has selected the match rule, and the match body applies. The match *body* contains statements that affect the subsequent handling of the request.

Multiple match rules are checked in order. Once the data in the request selects a match rule -- that is, the match rule expression evaluates to true -- no further match rules are checked against the request.

Equalizer makes a load balancing decision as follows:

1. If the request headers contain a cookie that specifies a server pool for the match rule, Equalizer sends the request to the server in the cookie.

   Otherwise:
2. Equalizer sends the request to the server pool specified in the match rule that is selected by the load balancing policy in effect for the match rule.

This process applies even if all the servers selected for the match rule are unavailable. In this case, when the match rule expression matches the request and all the servers in the match rule server list are unavailable, no reply is sent to the client. Eventually, the client sees a connection timeout.

If the match expression evaluates to *false*, then each subsequent match rule in the list of match rules for the virtual cluster is processed until a match occurs. All clusters have a Default Match rule, which always evaluates to *true* and which will use the entire set of servers for load balancing. The **Default Match** rule is always processed last.

Each cluster can have any number of match rules, and each match rule can have arbitrarily complex match expressions. Keep in mind that Equalizer interprets match rules for every Layer 7 cluster connection, so it is a good idea to keep match rules as simple as possible.

Match Rule Order

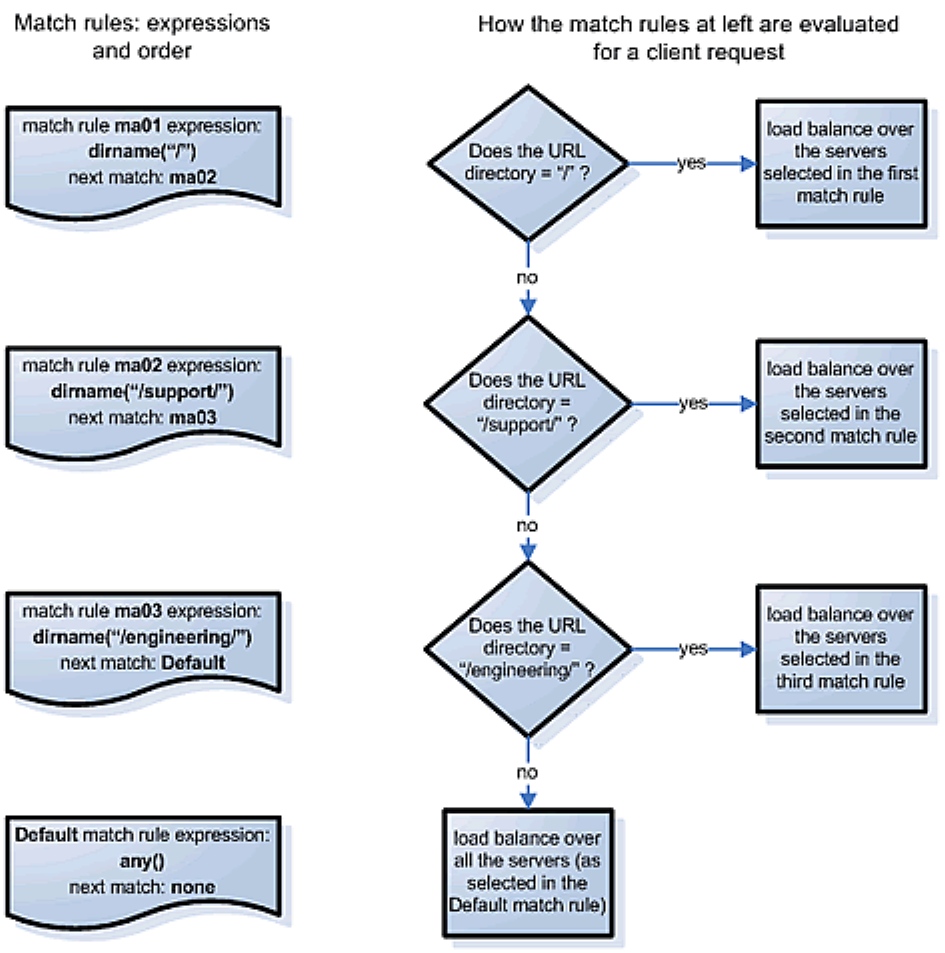
When you add more than one match rule to a cluster, the order in which the match rules are processed is important to system performance. Since processing a match rule requires system CPU and memory, the most efficient way of ordering match rules is from the most common case to the least common case. In this way, you ensure that the greatest number of client connections possible will process the first match rule and, if it matches the request, stop processing match rules for that request.

In other words, the goal is to load balance the highest possible number of requests according to the settings in the first match rule, which has the effect of reducing to a minimum the amount of match rule processing required for requests to that cluster.

This is best illustrated by an example. Let’s say you want to construct a set of match rules that achieves these goals:

- Direct all requests whose URL contains one of two specific directories to specific server pools. Assume these two directories are.../support and.../engineering.
- Of the two directories above, we expect more requests to contain.../support.
- Load balance requests whose URL *does not* contain a directory across all servers.
- We want to process requests that *do not* contain a directory the fastest, since we expect that 75% of requests to this cluster will NOT contain a directory in the URL.

The set of match rules that achieves this, their order, and how the match rules are evaluated, is described in the following figure.


Match rules: expressions and order
How the match rules at left are evaluated for a client request
match rule ma01 expression: dirname("/") next match: ma02
match rule ma02 expression: dirname("/support/") next match: ma03
match rule ma03 expression: dirname("/engineering/") next match: Default
Default match rule expression: any() next match: none
Does the URL directory = "/" ?
Does the URL directory = "/support/" ?
Does the URL directory = "/engineering/" ?
yes
no
load balance over the servers selected in the first match rule
load balance over the servers selected in the second match rule
load balance over the servers selected in the third match rule
load balance over all the servers (as selected in the Default match rule)

At left in the figure above are the expressions for the three match rules, shown in the order in which they are configured in the cluster. At right, the decision tree describes how the match rules are evaluated for every client request that comes into this cluster.

As described previously, the first match rule (**ma01**) is meant to match any request that does not have a directory in it. Since this is our most common case, match rule evaluation will stop after the first match rule is evaluated for the majority of incoming requests.

The second and third rules, **ma02** and **ma03**, match for specific directory names. We match for the most common directory name first, then the less common directory name.

Finally, if all three of the match rule expressions for **ma01**, **ma02**, and **ma03** fail to match an incoming request, then that request is load balanced across the server pool in the cluster using the options set on the cluster (and mirrored in the Default match rule).

Match Rule Expressions and Bodies

Match functions and operators are used to construct the *expression* parameter found in a match rule. The *expression* parameter selects the requests to be processed using the parameters specified in the remainder of the match rule.

Match Rule Expressions

Match rules consists of a *match expression* and a *match body,* which identifies the operations to perform if the expression is satisfied by the request. Match syntax is as follows:

```
match name {expression} then {body}
```

Each match has a name, which is simply a label. The name must follow the same restrictions as those for cluster names and server pool names. All match names within a cluster must be unique.

Match expressions affect the subsequent processing of the request stream using URI, host, or other information. They are made up of match functions, most of which are protocol-specific, joined by logical operators, optionally preceded by the negation operator, with sets of beginning and end parentheses for grouping where required. This may sound complex, and it can be, but typical match expressions are simple; it is usually best from a performance perspective to keep them simple.

The most simple match expression is one made up solely of a single match function. The truth value (*true* or *false*) of this expression is then returned by the match function. For example, a match function common to all Layer 7 protocols is the **any**() function, which always returns *true*, independent of the contents of the request data. So, the most simple match expression is:

```
any()
```

which will always result in the match rule being selected.

Use the logical NOT operator to invert the sense of the truth value of the expression. So, you can use the NOT operator to logically invert a match expression, as follows:

```
!expression
```

giving rise to the next simplest example:

```
!any()
```

which always evaluates to *false* and always results in the match rule not being selected.

With the addition of the logical OR (||) and logical AND (&&) operators, you can specify complex expressions, selecting precise attributes from the request, as in this:

```
!happy() || (round() && happy())
```

Match expressions are read from left to right. Expressions contained within parentheses get evaluated before other parts of the expression. The previous expression would match anything that was not happy or that was round and happy.

Unlike the previous example, match functions correspond to certain attributes in a request header.

For example, a request URI for a web page might look like this:

```
\Get /somedir/somepage.html http/1.1
Accept: text/html, text/*, *.*
Accept-Encoding: gzip
Host: www.website.com
User-Agent: Mozilla/4.7 [en] (Win98; U)
```

Various functions return true when their arguments match certain components of the request URI. Using the above request URI, for example, you could use several match functions:

- **pathname()** *returns true if its argument matches* /somedir/somepage.html
- **dirname()** *returns true if its argument matches* /somedir/
- **filename()** *returns true if its argument matches* somepage.html

Other functions can evaluate the contents of the Host header in the request URI above:

```
host (www.website.com)
host_prefix (www)
host_suffix (website.com).
```

Some function arguments can take the form of a regular expression[1]. Note that you cannot put regular expressions.

```
Matching regular expressions (using *_regex() functions) is many times more
processing-intensive than using other match functions. It is usually possible to
avoid using regular expressions by carefully crafting match expressions using
other functions. For example, the following regular expression match:
dirname_regex("(two|four|six|eight)")
Can be replaced by the more efficient:
dirname_substr("two") ||
dirname_substr("four") ||
dirname_substr("six") ||
dirname_substr("eight")
```

Match Bodies

[1]Regular expressions are specified according to IEEE Std 1003.2 (“POSIX.2”).

Match *bodies* specify the actions to take if the match expression selects the request. This is specified in the form of statements that provide values to variables used by the load balancer to process the request. The most common (and most useful) match body selects the set of servers (server pool) over which to apply the load balancing.

The **servers** assignment statement takes a comma-separated list of server names, which specifies the set of servers to be used for load balancing all requests that match the expression in the match rule. The reserved server names all and none specify respectively the set of *all* servers in the virtual cluster and *none* of the servers in the virtual cluster. If you do not assign servers, none will be available for load balancing; as a result, the connection to the client will be dropped.

In general, you can override most cluster-specific variables in a match body. One useful example of overriding variables is as follows:

```
    flags =!once_only;
```

which would load-balance across the specified server pool (which first must be defined in the virtual cluster) and also turn off the **once_only** flag for the duration of processing of that connection.

Match Rule Functions

Match rule *functions* generally test for certain strings or settings in the headers and URI of a client request. In the table below, we first discuss match rule functions that examine information in the request other than the URI, and then we discuss the URI related functions.

The following table lists the non-URI functions supported by Equalizer match rules:

any()	This function always evaluates to *true* -- that is, this function matches any incoming request.
client_ip(string)	This function evaluates to *true* only if the IP address of the client machine making the connection matches the *string* argument. The *string* can be a simple IP address (e.g., “192.168.1.110”), or an IP address in Classless Inter-Domain Routing (CIDR) notation (e.g., “192.168.1.0/24”). This function can be useful in restricting match expressions to a particular client or group of clients, which can aid in debugging a new match rule when a cluster is in production. Only the specified clients match the rule, leaving other clients to be handled by other match rules
debug_message(string)	This function always evaluates to *true*. It writes the *string* argument to the Event Log for the cluster (**View > Event Log**). This function can be logically ANDed and ORed with other functions to write debug messages. *Use this function for testing and debugging only. Do not use it in production environments, since it has a negative impact on performance.*
ignore_case()	This function always evaluates to *true*, and is intended to be used to apply the **ignore_case** flag for comparisons when it is *not set* on the cluster. When this function is ANDed with other functions, it has the effect of forcing case to be ignored for any comparisons done by the match rule.
observe_case()	This function always evaluates to *true*, and is intended to be used to override the **ignore_case** flag for comparisons when it is *set* on a cluster. When this function is ANDed with other functions, it has the effect of forcing case to be honored for any comparisons done by the match rule.
http_09()	This function takes no arguments and evaluates to *true* if the HTTP protocol used by the request appears to be HTTP 0.9. This is done by inference: if an explicit protocol level is absent after the request URI, then the request is considered HTTP 0.9.
method(string)	This function evaluates to *true* if the *string* argument exactly matches the Request Method (e.g., GET, POST, etc.) specified in the request. Note that by default Equalizer forwards packets to servers without determining whether or not the method specified in the request is valid (i.e., is a method specified in Section 9 of RFC2616). One use of the **method()** function is to be able to override this default behavior and prevent invalid requests from being forwarded to a server.
ssl2()	HTTPS only. This function evaluates to *true* if the client negotiated the encrypted connection using SSL version 2.0.
ssl3()	HTTPS only. This function evaluates to *true* if the client negotiated the encrypted connection using SSL version 3.0.
tls1()	HTTPS only. This function evaluates to *true* if the client negotiated the encrypted connection using TLS version 1.0.

Non-URI header match functions

See Match Bodies, for the headers that can be specified in these functions.

header_prefix(header, string)	This function evaluates to *true* if the selected *header* is present and if the string-valued argument *string* is a prefix of the associated header text.
header_suffix(header, string)	This function evaluates to *true* if the selected *header* is present and if the argument *string* is a suffix of the header text.
header_substr(header, string)	This function evaluates to *true* if the selected *header* is present and if the string-valued argument *string* is a sub-string of the associated header text.
header_regex(header, string)	This function evaluates to *true* if the selected *header* is present and if the string-valued argument *string*, interpreted as a regular expression, matches the associated header text.

In addition to the functions in the preceding table, a set of functions is provided that allows you to process requests based on the various components of a request's destination URI.

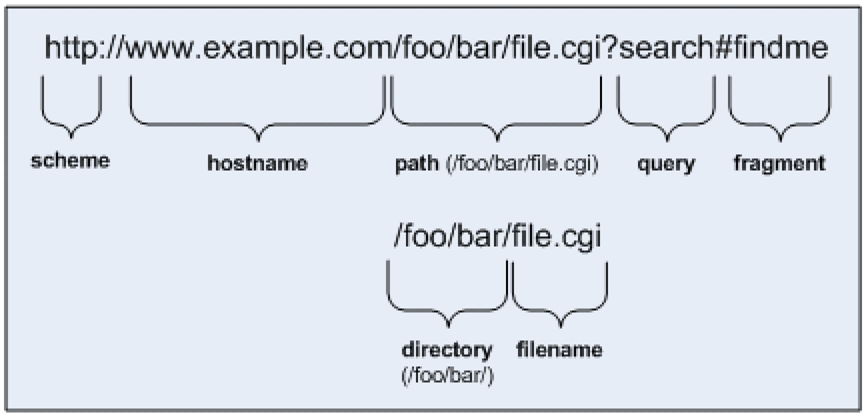
A URI has the following parts (as defined in RFC1808):

```
<scheme>://<hostname>/<path>;<params>?<query>#<fragment>
```

In addition, Equalizer further breaks up the <path> component of the URI into the following components:

```
<directory><filename>
```

The following figure illustrates how Equalizer breaks up a URI into the supported components:



Note that the following components of the URI do not have corresponding match functions:

- Match functions for the <scheme> component are not necessary, since a cluster must be configured to accept only one protocol: HTTP *or* HTTPS.
- Match functions for the optional <params> component are not provided. Use the **pathname*()** and **filename*()** functions to match characters at the end of the **path** and **filename** components.

- Match functions for the optional <fragment> component are not provided. The fragment portion of a URI is not transmitted by the browser to the server, but is instead retained by the client and applied after the reply from the server is received.

The following lists the URI matching functions that match text in the URI components shown.

URI Function	Description
host(string)	This function evaluates to *true* if the *string* argument exactly matches the hostname portion of the request. *In the case of HTTP 0.9, the host is a portion of the request URI. All other HTTP protocol versions require a Host header to specify the host, which would be compared to the string.*
host_prefix(string)	This function evaluates to *true* if the *string* argument is a prefix of the hostname portion of the URI path. The prefix of the hostname includes all text up to the first period; for eample, "www" in "www.example.com".
host_suffix(string)	This function evaluates to *true* if the *string* argument is a suffix of the hostname portion of the URI path. The suffix of the hostname includes all text after the first period in the hostname; for example, "example.com" in "www.example.com".
pathname(string)	This function evaluates to *true* if the *string* argument exactly matches the path component of the request URI.
pathname_prefix(string)	This function evaluates to *true* if the *string* argument is a prefix of the path component of the request URI.
pathname_suffix(string)	This function evaluates to *true* if the *string* argument is a suffix of the path component of the request URI.
pathname_substr(string)	This function evaluates to *true* if the *string* argument is a substring of the path component of the request URI.
pathname_regex(string)	This function evaluates to *true* if the *string* argument, interpreted as a regular expression, matches the path component of the request URI.
dirname(string)	This function evaluates to *true* if the *string* argument exactly matches the directory portion of the path component of the request URI. The path component is the entire directory path, including the trailing slash. For example, "/foo/bar/" is the directory portion of "/foo/bar/file.html".
dirname_prefix(string)	This function evaluates to *true* if the *string* argument is a prefix of the directory portion of the path component of the request URI. The leading slash must be included in the *string* (for example, "/fo" is a prefix of "/foo/bar/file.html").
dirname_suffix(string)	This function evaluates to *true* if the *string* argument is a suffix of the directory portion of the path component of the request URI. The trailing slash must be included in the *string* (for example, "ar/" is a suffix of the directory portion of "/foo/bar/file.html").
dirname_substr(string)	This function evaluates to *true* if the *string* argument is a substring of the directory portion of the path component of the request URI.
dirname_regex(string)	This function evaluates to *true* if the *string* argument, interpreted as a regular expression, matches the directory portion of the path component of the request URI.

URI Function	Description
filename(string)	This function evaluates to *true* if the *string* argument exactly matches the filename portion of the URI path. *This portion includes only the text after the last trailing path component separator (/), as that is considered part of the directory* (for example, "file.html" is the filename portion of "/foo/bar/file.html").
filename_prefix(string)	This function evaluates to *true* if the *string* argument is a prefix of the filename portion of the URI path.
filename_suffix(string)	This function evaluates to *true* if the *string* argument is a suffix of the filename portion of the URI path.
filename_substr(string)	This function evaluates to *true* if the *string* argument is a substring of the filename portion of the URI path.
filename_regex(string)	This function evaluates to *true* if the *string* argument, interpreted as a regular expression, matches the filename portion of the URI path.
query(string)	This function evaluates to *true* if the *string* argument exactly matches the (optional) query component of the request URI. The query, if present, appears in a URI following a question mark (?). The syntax of a query is application specific, but generally is a sequence of key/value pairs separated by an ampersand (&).
query_prefix(string)	This function evaluates to *true* if the *string* argument is a prefix of the query portion of the URI path.
query_suffix(string)	This function evaluates to *true* if the *string* argument is a suffix of the query portion of the URI path.
query_substr(string)	This function evaluates to *true* if the *string* argument is a substring of the query portion of the URI path.
query_regex(string)	This function evaluates to *true* if the *string* argument, interpreted as a regular expression, matches the query portion of the URI path.

Match Rule Operators

Match Rule Operators are as follows:

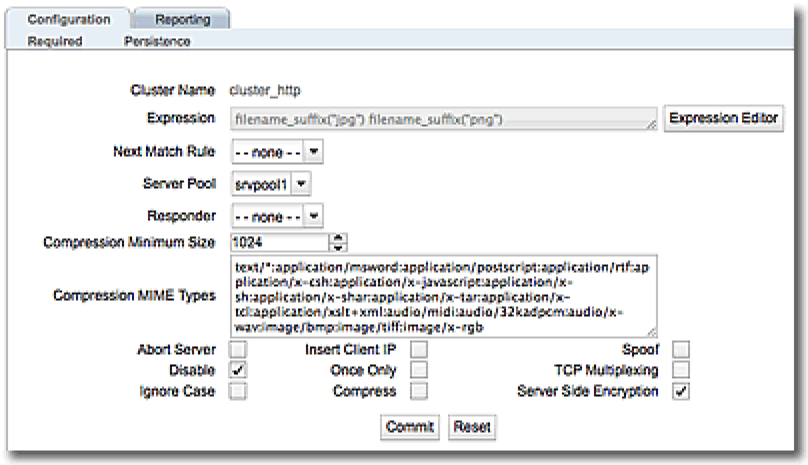
- **II -** logical OR operator
- **&& -** logical AND operator
- **! -** logical NOT operator
- **()** - used to group functions and operators

Match Rule Definitions

Match *rules* are defined in the file `/var/eq/eq.conf` with the definition of the cluster to which the match rule applies. A match rule as it appears in `eq.conf` looks like the following example:

```
match ma01 {
client_ip("10.0.0.19")
} then {
flags =!spoof;
srvpool = sv_01;
}
```

In this example (the match rule is named “ma01”), the match function, **client_ip**, has an argument that matches all requests from IP address 10.0.0.19, which are all sent to server sv_01. Additionally, this rule disables the **spoof** flag (that is, when the connection is made to the server, the server sees a connection to the Equalizer, not to the client). This is displayed as follows:



Cluster Name	cluster_http
Expression	filename_suffix("jpg") filename_suffix("png")
Next Match Rule	- - none - -
Server Pool	srvpool1
Responder	- - none - -
Compression Minimum Size	1024
Compression MIME Types	text/*:application/msword:application/postscript:application/rtf:application/x-csh:application/x-javascript:application/x-sh:application/x-shar:application/x-tar:application/x-tcl:application/xslt+xml:audio/midi:audio/32kadpcm:audio/x-wav:image/bmp:image/tiff:image/x-rgb

Abort Server	☐	Insert Client IP	☐	Spoof	☐
Disable	✓	Once Only	☐	TCP Multiplexing	☐
Ignore Case	☐	Compress	☐	Server Side Encryption	✓

The **Expression** field shows the expression that is evaluated against the incoming request. If the expression evaluates to *true*, the **Server Pool** field specifies the "pool" of servers that will be used to satisfy the incoming request, as well as the options that will be set for the request.

Refer to "Managing Server Pools" on page 476

Match Rule Expression Examples

A match rule *expression* must be specified in double quotes, so any quotes used in a function to delineate strings must be escaped with a backslash character (\), as in this example that matches all client requests with a source IP on the 10.10.10/24 network:

```
    expression “client_ip(\“10.10.10/24\”)”
```

Functions can be negated using the “!” operator. To change the above example to match all client requests with a source IP *not* on the 10.10.10/24 network, use this expression:

```
expression “!client_ip(\“10.10.10/24\”)”
```

Functions can be combined using the logical operators shown in the previous section. For example, to match a client request for any file with two different file suffixes, you could use an expression like this:

```
expression “filename_suffix(\“jpg”) or filename_suffix(\“gif”)”
```

Functions and operators can be grouped using parentheses to create complex expressions. For example, to match a client request with a source IP on the 10.10.10/24 network and a URI whose filename suffix is *not* “jpg” or “gif”, use the following expression:

```
expression “client_ip(\“10.10.10/24\”) and!(filename_suffix(\“jpg”) or filename_
suffix(\“gif”))”
```

Match Rule Expression Notes

Observe the following when constructing match rule expressions:

Match Rule Behavior When Server Status is Not "Up"

When a match rule expression matches a client request, the request is load balanced using the server pools, parameters, and flags specified in the match rule. The server pools specified in the match rule may be in a number of “states” that affect the load balancing behavior: the servers within the sever pools may be up or down, and may have one or both of the **quiesce** and **hot spare** options enabled.

server up	The request is routed to the selected server.
up/quiesce enabled	The request is routed to the selected server.
up/hot spare enabled	The request is routed to the selected server.
server down	If no Responder is selected in the match rule, then the request is sent to the selected server and, eventually, the client times out. If a Responder is selected, the Equalizer sends the configured response to the client.

The reason match rules behave as shown above is because the purpose of a match rule is to send a request that matches an expression to a particular server that can (presumably) better satisfy the request. In some cases, sending the request to a particular server may be required behavior for a particular configuration.

With this in mind, it does not make sense to skip a match rule because the server (or servers) named in the rule are down, hot spared, or quiesced -- rather, since the server in the rule is presumably critical to satisfying the request, it makes sense to route the request to the (for example) down server, and have the client receive an appropriate error -- so that the request can be retried.

If we instead were to skip a match rule because, for example, the server selected by the match rule is down, the request would be evaluated by the next match rule -- or the default match rule. The request, therefore, could potentially be sent to a server in the cluster that does not have the requested content. This means that the client would receive a “not found” error, instead of an error indicating that the appropriate server is not currently available.

Considering Case in String Comparisons

String comparisons performed by match functions honor the setting of the **ignore case** cluster parameter: if it is set on the cluster (the default), then all match rule functions used for that cluster are case insensitive; that is, the case of strings is ignored. For example, the string “ab” will match occurrences of “ab”, “Ab”, “aB”, and “AB”. If **ignore case** is *not* set on the cluster, then all string comparisons are by default case sensitive (the string “ab” will match only “ab”).

To override the **ignore case** flag setting on the cluster for a match function or block of functions, you must logically AND the **observe_case()** or **ignore_case()** functions with the match function or block. For example, if **ignore case** is set on the cluster, you would use the following **expression** to force the **header_substr()** function to make case sensitive string comparisons:

```
    (observe_case() and header_substr(\"host\", \"MySystem\"))
```

Regular Expressions

Some match functions have *prefix*, *suffix*, *substr*, or *regex* variants. The *regex* variants interpret an argument as a regular expression to match against requests. Regular expressions can be very costly to compute, so use the *prefix*, *suffix*, or *substr* variants of functions (or Boolean combinations of prefix and suffix testing), rather than the *regex* function variants, for best performance. For example, the following regular expression match:

```
dirname_regex(\"(two|four|six|eight)\")
```

Can be replaced by the more efficient:

```
dirname_substr(\"two\") OR

dirname_substr(\"four\") OR

dirname_substr(\"six\") OR

dirname_substr(\"eight\")
```

Note that Equalizer match rule expressions support POSIX regular expression syntax only.

Supported Headers

All of the **header_***match functions take a **header** argument, which selects the header of interest. If this header is not present in the request, the match function evaluates to *false*.

Although HTTP permits a header to span multiple request lines, none of the functions match text on more than one line. In addition, Equalizer will only parse the first instance of a header. If, for example, a request has multiple **cookie** headers, Equalizer will only match against the first **cookie** header in the request.

Any text that you enter will be accepted as the header text.

HTTPS Protocol Matching

Equalizer permits the construction of virtual clusters running the HTTPS protocol. HTTPS is HTTP running over an encrypted transport, typically SSL version 2.0 or 3.0 or TLS version 1.0. All of the functions available for load balancing HTTP clusters are available for HTTPS clusters. In addition, there are some additional match functions [ssl2(), ssl3(), and tls1()], that match against the protocol specified in an HTTPS request.

Supported Characters in URIs

The characters permitted in a URI are defined in RFC2396. Equalizer supports all characters defined in the standard for all Match Functions that have a URI as an argument. Note in particular that the ASCII space character is not permitted in URIs -- it is required to be encoded by all conforming browsers as “%20” (see Section 2.4 of RFC2396).

Match Rules, the Once Only Flag, and Cookies

Since multiple client requests may be received on a single TCP/IP connection, Equalizer has a flag (**once only**) that specifies whether to check the headers in every request received on a connection, or to load balance based solely The **once only** flag is a cluster parameter on the **Networking** tab. When using Match Rules, it is usually desirable to turn off the **once only** flag for the cluster so that Equalizer matches against each individual request in a connection, not just the first one.

You can also enable or disable **once only** flag in the match rule **Configuration** screen (tab), to override the setting on the cluster for any request that matches that rule. For example, if **once only** is enabled on a cluster and disabled on a match rule, any request that matches that match rule's expression will be load balanced as if **once only** were disabled on the cluster.

The following table shows how the setting of once only affects load balancing when a match rule hit occurs:

match rule hit on...	**once only disabled**	**once only enabled**
...the first request on a connection	If the request headers contain a cookie specifying a server in the match rule's server list, send the request to the server in the cookie. Otherwise, send the request to the server in the match rule's server list that is selected by the load balancing policy in effect for the match rule.	Same as at left.
...second and subsequent requests on the same connection	Same as above.	If the request headers contain a cookie specifying a server in the match rule's server list, send the request to the server in the cookie. Otherwise, send the request to the server that was selected by the first request.

Note that Equalizer always honors a cookie that specifies a server in the match rule's server pool list, regardless of the setting of the **once only** flag: the request is sent to the server pool specified by the cookie. If, however, the cookie specifies a server pool that is not in the match rule's server list, the cookie is ignored.

Using Responders in Match Rules

Responders are used to send automated responses to clients when all the server pools in a match rule are down. See "Modifying a Responder" on page 618 for a complete description of Responders as well as examples of using Responders in Match Rules.

Managing Match Rules

The EQ/OS 10 Administration Interface allows you to create and modify match rules, without requiring a detailed knowledge of the configuration language syntax used in the *eq.conf* file. The interface validates match rules before saving them so that all saved rules are syntactically correct. For this reason, we recommend you use the interface to create and edit match rules, rather than editing the configuration file.

The interface does *not*, however, test the behavior of match rules. Match rules must be tested against a flow of incoming requests in order to determine if the behavior of the rule is what you expect.

Before constructing a match rule, you should first understand the general concepts of match rules covered in "Match Rule Expressions and Bodies" on page 431.

In the Match Rule descriptions herein, instructions are provided for using the GUI first, followed by instructions for accomplishing the same task using the CLI. Refer to "Working in the CLI" on page 141 for details on using the CLI commands.

Displaying Match Rules

On the GUI, click on a cluster name in the navigation pane and then click on any of the Match Rules associated with any of the HTTP or HTTPS clusters to display the match rules defined for that cluster.

On `eqcli`, enter the following:

```
eqcli > cluster clname match
```

In the example below the Match Rules on the cluster "SP-fe_http" are displayed:

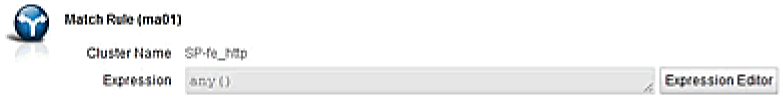
```
eqcli cl-SP-*> show match
Name Server Pool Responder Expression
nopersist serverool1 responder1 *
images *
test *
ma01 adp_tcp *
New_Match_Rule *
```

Default Match Rule

All Layer 7 clusters created via the Equalizer Administration Interface start with a single match rule (named **Default**) that matches all requests and selects all servers.

```
match Default {
any()
} then {
servers = all;
}
```

When displayed `any()` appears in the Expression field in the GUI as shown below.



Match Rule (ma01)

Cluster Name SP-fe_http

Expression any()

The default rule specifies that all server pools defined in the cluster should be used for load balancing the request, and that all flag settings for the request will be inherited from the cluster flag settings.

Creating a New Match Rule

Match rules can be created using either the CLI or the GUI.

Parameters

The table below shows the parameters, values, and flags used in the configuration of match rules

GUI Parameter (CLI Parameter)	Description
Expression **(`expression`)**	Refer to "Match Rule Expression Notes" on page 440.
Next Match Rule **(`nextmatch`)**	The **Next Match Rule** field determines the order of processing. For example, if you were to configure a Match Rule 2 with a **Next Match Rule** parameter of Match Rule 1, it would be place *before* Match Rule 1 in the order of processing.
Server Pool **(`srvpool`)**	The **ServerPool** field determines the server pool to which a match rule applies its specified conditions and parameters.
Responder **(`resp`)**	The **Responder**field allows you to specify an automatic responder for client requests that match this rule when none of the servers selected in the rule are available. The responder must already be configured. For a description of responders as well as examples of using responders in match rules, see "Adding a Responder" on page 616.
Compression Minimum Size **(`compress_min`)** **(ADCs with Hardware Acceleration)**	The minimum file size in bytes required for GZIP compression, if enabled. Equalizer uses GZIP to compress the payload (or content) of the server response before sending it back to the client. This is typically done for 2 reasons: faster client response and better user experience. In addition. less ISP bandwidth is used in sending smaller files back to clients. Files smaller than the minimum specified are not compressed. Default:1024 bytes.cceleration)
Compression MIME Types **(`compress_types`)** **(ADCs with Hardware Acceleration)**	Specifies the mime-types that will be compressed when the **Compress** option is enabled for the match rule. The value of this parameter is a string (maximum length: 512 characters) with valid mime-type names separated by a colon (:). The default compress mime-types string specifies the following mime-types
Flags	
Spoof **(`spoof`)**	**Spoof**causes Equalizer to spoof the client IP address when Equalizer routes a request to a server in a virtual cluster; that is, the IP address of the *client* is sent to the server, not the IP address of the Equalizer. This option is on by default. If you disable this option, the server receiving the request will see the Equalizer's address as the client address because the TCP connection to the client is terminated when the request is routed. When spoof is enabled, the server pool in the cluster must use the Equalizer as the default gateway for routing.
Abort Server **(`abort_server`)**	By default, when a client closes a connection, Equalizer waits for a response from the server before closing the server connection. If this flag is enabled, Equalizer will not wait for a response before closing the connection to the server; instead it sends a TCP RST (reset) to the server when the client closes the connection.
Ignore Case **(`ignore_case`)**	This function always evaluates to *true*, and is intended to be used to apply the **Ignore Case**flag for comparisons when it is *not set* on the cluster. When this function is ANDed with other functions, it has the effect of forcing case to be ignored for any comparisons done by the match rule.

GUI Parameter (CLI Parameter)	Description
Insert Client IP (insert_client_ip)(HTTPS only)	When this flag is enabled, Equalizer inserts an **X-forwarded-for:**header with the client's IP address into all client requests before they are sent to the server. This flag is *disabled* by default for HTTP clusters and *enabled* by default for HTTPS clusters.
Once Only (once_only)	Limits Equalizer to parsing headers (and executing match rules) for only the first request of any client making multiple requests across a single TCP connection. This option is off by default: meaning that Equalizer will parse the headers of every client request.
Disable (disable)	Enable this flag to disable this match rule without deleting it. This can be useful when testing new match rules.
TCP Multiplexing (tcp_mux)	Enables TCP multiplexing for a cluster. TCP multiplexing must also be enabled on at least one server instance in the server pool assigned to the cluster (or one of its match rules).
Compress (compress)	When this option is enabled, Equalizer automatically detects requests to the cluster/match rule from compression-capable browser clients and performs GZIP compression on all responses sent to that client. The cluster IP address will not accept requests when this flag is enabled. When enabled, hardware compression will be used on ADCs with compression hardware on board and software compression will be used on ADCs without compression hardware.
Server Side Encryption (sse)	Enables the **Server Side Encryption** option for the Match Rule. By enabling this option, Equalizer will encrypt packets to back end servers using the SSL/TLS specifications in the Server Side Encryption Global Parameters (See "Server Side Encryption" on page 276)

GUI Parameter (CLI Parameter)	Description
Expression (expression)	Refer to "Match Rule Expression Notes" on page 440.
Next Match Rule (nextmatch)	The **Next Match Rule** field determines the order of processing. For example, if you were to configure a Match Rule 2 with a **Next Match Rule** parameter of Match Rule 1, it would be place *before* Match Rule 1 in the order of processing.
Server Pool (srvpool)	The **ServerPool** field determines the server pool to which a match rule applies its specified conditions and parameters.
Responder (resp)	The **Responder**field allows you to specify an automatic responder for client requests that match this rule when none of the servers selected in the rule are available. The responder must already be configured. For a description of responders as well as examples of using responders in match rules, see "Adding a Responder" on page 616.

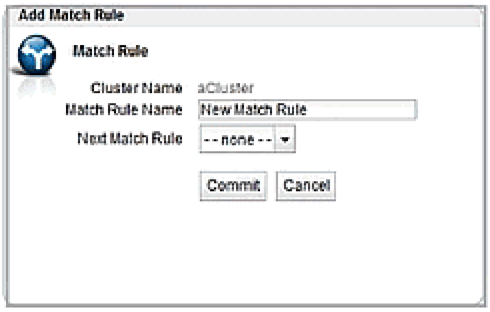
GUI Parameter (CLI Parameter)	Description
Compression Minimum Size (`compress_min`) **(ADCs with Hardware Acceleration)**	The minimum file size in bytes required for GZIP compression, if enabled. Equalizer uses GZIP to compress the payload (or content) of the server response before sending it back to the client. This is typically done for 2 reasons: faster client response and better user experience. In addition. less ISP bandwidth is used in sending smaller files back to clients. Files smaller than the minimum specified are not compressed. Default:1024 bytes.cceleration)
Compression MIME Types (`compress_types`) **(ADCs with Hardware Acceleration)**	Specifies the mime-types that will be compressed when the **Compress** option is enabled for the match rule. The value of this parameter is a string (maximum length: 512 characters) with valid mime-type names separated by a colon (:). The default compress mime-types string specifies the following mime-types
Flags	
Spoof (`spoof`)	**Spoof**causes Equalizer to spoof the client IP address when Equalizer routes a request to a server in a virtual cluster; that is, the IP address of the *client* is sent to the server, not the IP address of the Equalizer. This option is on by default. If you disable this option, the server receiving the request will see the Equalizer's address as the client address because the TCP connection to the client is terminated when the request is routed. When spoof is enabled, the server pool in the cluster must use the Equalizer as the default gateway for routing.
Abort Server (`abort_server`)	By default, when a client closes a connection, Equalizer waits for a response from the server before closing the server connection. If this flag is enabled, Equalizer will not wait for a response before closing the connection to the server; instead it sends a TCP RST (reset) to the server when the client closes the connection.
Ignore Case (`ignore_case`)	This function always evaluates to *true*, and is intended to be used to apply the **Ignore Case**flag for comparisons when it is *not set* on the cluster. When this function is ANDed with other functions, it has the effect of forcing case to be ignored for any comparisons done by the match rule.
Insert Client IP (`insert_client_ip`)(HTTPS only)	When this flag is enabled, Equalizer inserts an **X-forwarded-for:**header with the client's IP address into all client requests before they are sent to the server. This flag is *disabled* by default for HTTP clusters and *enabled* by default for HTTPS clusters.
Once Only (`once_only`)	Limits Equalizer to parsing headers (and executing match rules) for only the first request of any client making multiple requests across a single TCP connection. This option is off by default: meaning that Equalizer will parse the headers of every client request.
Disable (`disable`)	Enable this flag to disable this match rule without deleting it. This can be useful when testing new match rules.
TCP Multiplexing (`tcp_mux`)	Enables TCP multiplexing for a cluster. TCP multiplexing must also be enabled on at least one server instance in the server pool assigned to the cluster (or one of its match rules).

GUI Parameter (CLI Parameter)	Description
Compress **(`compress`)**	When this option is enabled, Equalizer automatically detects requests to the cluster/match rule from compression-capable browser clients and performs GZIP compression on all responses sent to that client. The cluster IP address will not accept requests when this flag is enabled. When enabled, hardware compression will be used on ADCs with compression hardware on board and software compression will be used on ADCs without compression hardware.
Server Side Encryption **(`sse`)**	Enables the **Server Side Encryption** option for the Match Rule. By enabling this option, Equalizer will encrypt packets to back end servers using the SSL/TLS specifications in the Server Side Encryption Global Parameters (See "Server Side Encryption" on page 276)

Creating Match Rules using the GUI

Proceed with the following to add a match rule to a virtual cluster using the GUI:

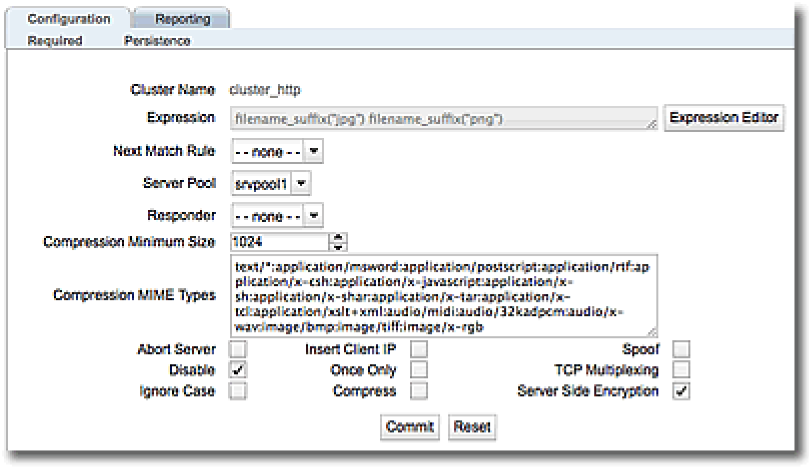
1. Log into the GUI using a login that has add/del access for the cluster. See Logging in on page.
2. In the navigation pane on the left, right-click the name of the Layer 7 cluster to which you want to add a match rule, and select **Add New Match Rule** to display the following:



Cluster Name: aCluster
Match Rule Name: New Match Rule
Next Match Rule: -- none --

3. Enter a name for the new rule in the **Match Rule Name**field. All match names within a cluster must be unique.
4. Make a selection for the **Next Match Rule** using the drop-down list. When you select **Next Match Rule,**the new match rule you are creating will be placed before the **Next MatchRule** and will be evaluated in that sequence in load balancing.
5. Click **Commit** when are finished. The **Configuration** screen (tab) will be displayed as shown below.

Note - If you do not enable a check box for at least one server pool, *Equalizer will drop the connection for any request that matches the rule*. You must also associate a server pool with the match rule on the **Configuration** screen (tab).



Cluster Name: cluster_http
Expression: filename_suffix("jpg") filename_suffix("png")
Next Match Rule: -- none --
Server Pool: srvpool1
Responder: -- none --
Compression Minimum Size: 1024
Compression MIME Types: text/*:application/msword:application/postscript:application/rtf:application/x-csh:application/x-javascript:application/x-sh:application/x-shar:application/x-tar:application/x-tcl:application/xslt+xml:audio/midi:audio/32kadpcm:audio/x-wav:image/bmp:image/tiff:image/x-rgb

Abort Server	☐	Insert Client IP	☐	Spoof	☐
Disable	✓	Once Only	☐	TCP Multiplexing	☐
Ignore Case	☐	Compress	☐	Server Side Encryption	✓

6. Use the Expression Editor to build your match expression. Refer to"Match Rule Expression Examples" on page 439 for details on using this feature.

7. Use the Server Pool drop down list to select a Server Pool to direct Layer 7 traffic if it complies with the match rule conditions specified. Refer to "Managing Server Pools" on page 476 for instructions on configuring Server Pools.
8. Configure the other parameters for the **Match Rule**as necessary. Use the table above for parameters, values, and flags for the configuration of the match rule.
9. Click on the **Commit** button after making changes to the settings.

6. Use the Expression Editor to build your match expression. Refer to"Match Rule Expression Examples" on page 439 for details on using this feature.
7. Use the Server Pool drop down list to select a Server Pool to direct Layer 7 traffic if it complies with the match rule conditions specified. Refer to "Managing Server Pools" on page 476 for instructions on configuring Server Pools.
8. Configure the other parameters for the **Match Rule**as necessary. The following table describes each of the selections.Changing these parameters will override the cluster setting.
9. The ordering of match rules is important, as they are processed from first to last until one of them evaluates to *true*, at which time the match body is processed. The initial match expression of a new rule, any() is one that will always evaluate to *true*, meaning that this match rule will always be selected. It is good practice to be cautious when adding new match rules to ensure that all the traffic to a cluster does not get mishandled. Use the **Disable** flag to skip a match rule that is still being developed.
10. Click on the **Commit** button to Commit the parameter selections.

Adding Match Rules using the CLI

1. Log into the CLI using a login that has add/del access for the cluster. (See "Starting the CLI" on page 142.)
2. At the eqcli prompt enter the cluster name followed by "match *maname*". In the example below a Match Rule **test** is added to the Layer 7 cluster **Sp-fe_http**.

```
Match Rule Name : test
Next Match Rule : ma01
Cluster Name : SP-fe_http
Server Pool :
Responder :
Cookie Path :
Cookie Domain :
Cookie Scheme : 0
Cookie Age : 0
Cookie Generation : 0
Flags : disable
Expression :
any()
```

```
Match Rule Name : test
Next Match Rule : ma01
Cluster Name : SP-fe_http
Server Pool :
Responder :
Cookie Path :
Cookie Domain :
Cookie Scheme : 0
Cookie Age : 0
Cookie Generation : 0
Flags : disable
Expression :
any()
```

3. Assign a Server Pool to the newly created Match Rule by entering:

```
eqcli cl-clname-ma-maname> srvpool spname
```

4. Add or remove **Responder**, **CookiePath**, **CookieDomain**, **CookieScheme**, **CookieAge** and **CookieGeneration** and **Flags** using the Parameters table above as a reference.
5. Configure the Match Expressions using the following at the eqcli prompt. Descriptions of the Expressions are provided in "Match Rule Functions" on page 435.

```
eqcli cl-clname-ma-maname> expression string
```

Modifying a Match Rule

To edit a match rule using the GUI, follow these steps:

1. Log into the GUI using a login that has write access for the cluster (See "Logging In" on page 238).
2. In the navigation pane on the left click the name of the match rule to be changed.
3. Make the desired changes to the match rule, as shown in the procedure in the previous section, starting at Step 5.

To edit a match rule using eqcli follow these steps:

1. Log into eqcli using a login that has add/del access for the cluster (See "Starting the CLI" on page 142)
2. Make the desired changes using eqcli as shown in the procedures beginning with step 1.

Removing a Match Rule

To delete a match rule using the GUI, follow these steps:

1. Log into the GUI using a login that has add/del access for the cluster (See "Logging In" on page 238).
2. In the navigation pane on the left, right-click the name of the match rule to be deleted and select **Delete Match Rule**.
3. Click **delete** to confirm that you want to delete the match rule.

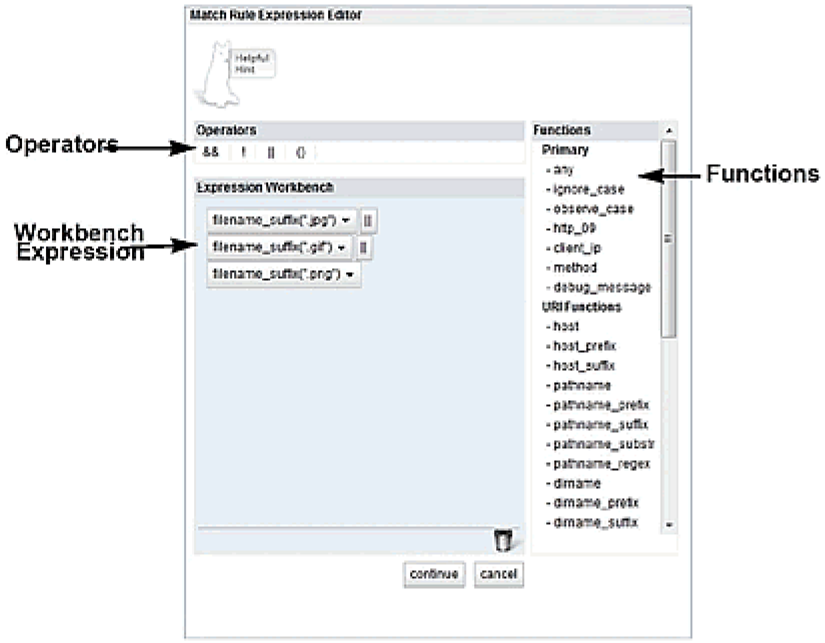
To delete a match rule using eqcli, follow these steps:

1. Log into eqcli using a login that has add/del access for the cluster (See "Starting the CLI" on page 142).
2. Enter the following at the eqcli prompt:

```
eqcli > cluster clname no match maname
```

Using the Match Rule Expression Editor

The **Match Rule Expression Editor** shown below is a feature of the GUI that allows the user an easy method of building Match Expressions. As described in "Match Rule Expressions and Bodies" on page 431, Match Expressions are made up of match functions, most of which are protocol-specific, joined by logical operators, optionally preceded by the negation operator, with sets of beginning and end parentheses for grouping where Match Bodies required. The Expression editor allows the user to drag and drop functions and operators to build the desired expressions.



The **Match Rule Expression Editor** is separated into 3 panes.

- The **Operators** pane displays the available operators:

"$$" is used for the logical AND operator.

"!" is used for the logical NOT operator.

"II" is used for the logical OR operator.

"()" is used to group functions and operators

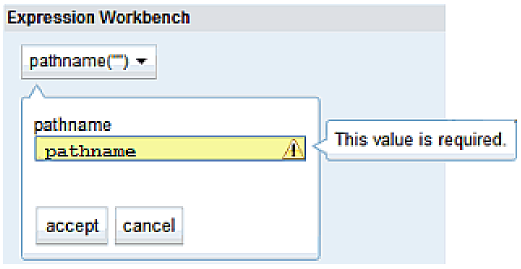
- The **Functions** (refer to "Match Rule Functions" on page 435) are displayed on the right pane displays a list of all of the available functions.
- The **Expression Workbench** is the work area used to build the expressions.

Operating within the Expression Editor

You can drag **Functions** and **Operators** into the **Expressions Workbench.** If you drag a new element onto the top of an existing element, the new element will be place before the existing element. If you drag the new element onto the bottom of en existing element, the new element will be placed after the existing element.

Elements can also be dragged moved within the **Expressions Workbench** or to the trash .

in many cases, **Functions** require a value such as shown below where the input of a path is required. Click on the **Function** to display the input field to enter the required details. Click on the **Accept** button to add the details to the **Function** or **Cancel** to discard the details.



pathname

This value is required.

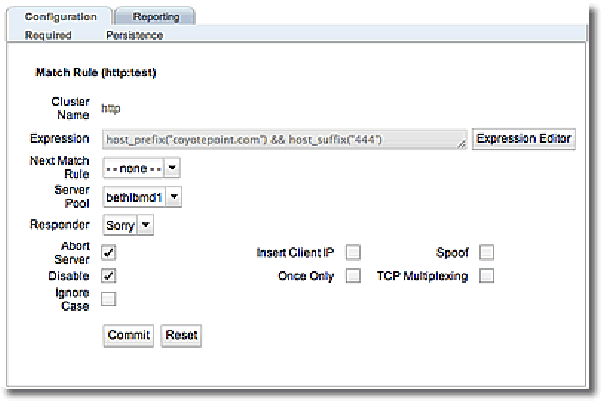
Clicking on the **continue** or **cancel** button will close the **Expression Editor.**

Clicking on the **Reset** button will remove all of your configured parameters and return to the default screen.

Clicking on the **Commit** button will assign all of your match rule configurations to the cluster.

The figure below shows an example of a completed Match Rule configuration. In this example a match rule is configured so that the incoming URL will be analyzed for the file extensions*.jpg*,*.gif* and*.png*. It this suffix is found, the incoming graphical files will be directed by Equalizer to a **Server Pool** called **Images.**

> **Note** - If a new **Server Pool** is required in the match rule for redirection by Equalizer it must be configured prior to creating the match rule. Refer to "Managing Server Pools" on page 476 to configure a new **Server Pool** if a new one is required.



Match Rule (http:test)

Cluster Name	http
Expression	host_prefix("coyotepoint.com") && host_suffix("444")
Next Match Rule	-- none --
Server Pool	bethlbmd1
Responder	Sorry
Abort Server	✓
Disable	✓
Ignore Case	☐
Insert Client IP	☐
Once Only	☐
Spoof	☐
TCP Multiplexing	☐

Example Match Rules

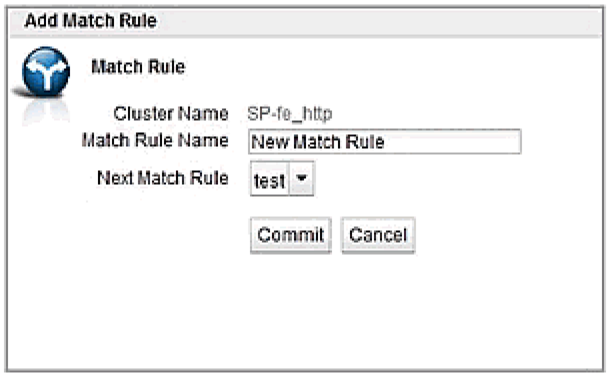
The Related Topics navigate to examples of how to create a few of the most commonly used types of match rules.

Parsing the URI Using Match Rules

In this example, we want to direct requests to a pool of specific server pools based on the hostname used in the URI contained in the request. We want all requests for URIs that start with "support" to go to one server pool, and all other requests that do *not* match this rule to be load balanced across all server pools in the cluster.

To do this, we will construct one match rule that parses the URI; if the URI contains the string **"support"**, it forwards the request to the server pool **sv_support**. For this example, we assume that a cluster with server pools has already been defined.

1. Log into the GUI using a login that has add/del access for the cluster.
2. In the left frame, right-click the name of the Layer 7 cluster to which you want to add the rule, and select **Add Match Rule**. The **Add Match Rule** dialog appears:
    a. Type a name into the **Match Rule Name** field. In this case **New Match Rule** was added.
    b. Select the **Next Match Rule** from the drop-down list to determine the placement of **New Match Rule** in the order of processing. In this case **test** was selected.


Add Match Rule

Match Rule

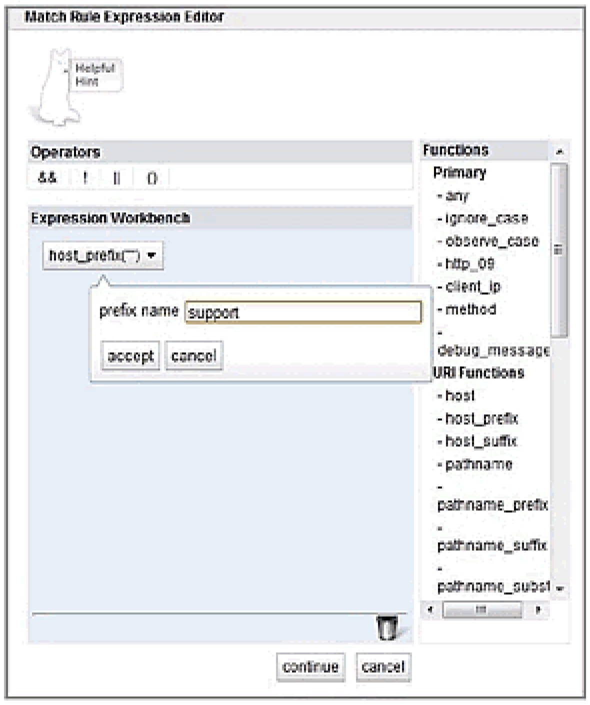
Cluster Name: SP-fe_http
Match Rule Name: New Match Rule
Next Match Rule: test

    c. Click on **Commit**. If **test** was selected as the **Next Match Rule** it will appear beneath **Test** in the navigation pane on the left.

The match rule is created, added to the navigation pane on the left, and its **Configuration** tab is opened. Refer to "Using the Match Rule Expression Editor" on page 455 for further descriptions on using the Expression Editor.

3. Click on the **Expression Editor** button and drag and drop **host_prefix** from the drop-down box in the **Functions** pane into the **Expression Workbench** pane.

4. Type "**support**" into the **hostname prefix** text box as follows:



5. Click on **accept** after entering "**support**" and then click on the **continue** button at the bottom of the **Expression Editor** to save the expression.

Now, all requests for URIs that start with "**support**" should go to the **sv_support** server pool, and all other requests that do *not* match this rule to be load balanced across all server pools in the cluster.

Changing Persistence Settings Using Match Rules

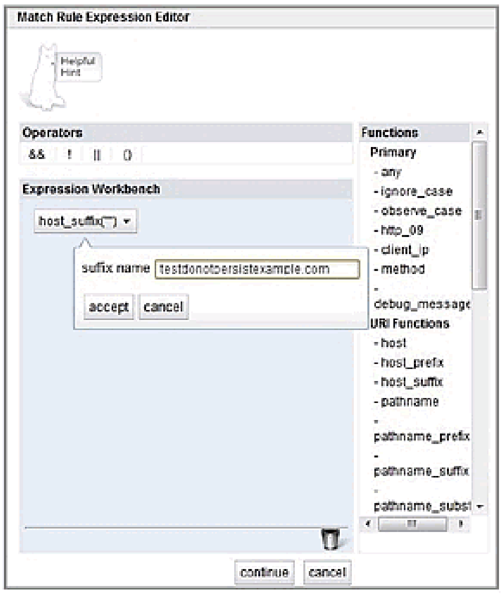
By default, a client request that matches a match rule expression is load balanced using the same load balancing parameters and options that are currently set on the cluster. The following describes how to change load balancing parameters and flags in a match rule.

For example, persistent connections to server pools are enabled by the **Persist** cluster flag, which is enabled by default when you create a cluster. Let's assume that you only want to *disable* persistence for incoming requests that have a URI containing a hostname in the following format:

```
xxx.testdonotpersistexample.com
```

We'll use the **host_suffix()** match rule function to test for the above hostname format. For this example, we assume that a cluster with three server pools has already been defined. We will construct a match rule that will turn off **Persist** for any request that contains the host suffix "**testdonotpersistexample.com**"; this request will be balanced across all of the server pools in the cluster.

1. Log into the GUI using a login that has add/del access for the cluster.
2. In the navigation pane on the left, right-click the name of the Layer 7 cluster to which you want to add the rule, and select **Add Match Rule.** The **Add Match Rule** dialog appears:
   a. Type **nopersist** into the **match name** text box.
   b. Select the server pool that this new rule will *precede* using the **Next Match Rule** drop-down list and click on **Commit.** The new rule will appear on the navigation tree in within the cluster from which is was created.
   c. On the match rule **Configuration** screen (tab) select the **Server Pool** that will be used for load balancing with the **Persist** checkbox disabled.
3. Click on the **Expression Editor** button to display the Expression editor.


Match Rule Expression Editor

suffix name: testdonotpersistexample.com

a. Leaving the **any()** expression in place, drag and drop the **host_suffix** from the **Functions** pane to the **Expression Workbox** *beside the* **any()** *expression.*

b. Type "**testdonotpersisteexample.com**" into the **hostname suffix()** function. The new expression should appear as follows.

c. Click on **continue.**

4. Uncheck the **Persist** checkbox and **Disable** checkboxes on the **Configuration** tab.
5. Click on **Commit** to save your changes to the **nopersist** rule.

You have now *disabled* persistence for incoming requests that have a URI containing the hostname "testexample.com".

Using Persistence with Match Rules

When a match rule is configured you can specify that persistence methods for that match rule -- which supercede those the persistence method specified for a cluster. This is the persistence type to be used when the match rules conditions are met. For example, if you configured a match rule expression to redirect requests to Server A based on the criteria configured in an expression, you can also configure the persistence type to be used when that criteria is met.

To configure persistence with match rules select a configured match rule on the left navigational pane of the GUI. Select the **Persistence** tab to display the configuration screen. It is configured the same as the configuration of HTTP and HTTPS cluster persistence.

Changing the Spoof (SNAT) Setting Using Match Rules

By default, Equalizer uses the client IP address as the source address in the packets it forwards to server pools, and then translates the server IP in server responses to Equalizer’s cluster IP. This is commonly called a *Half-NAT* configuration, since Equalizer is *not* performing Network Address translation (or NAT) on client requests. Because the server pools behind Equalizer see the source IP of the client, the server pools need to be configured to route client requests back through Equalizer -- either by making Equalizer the default. This behavior is controlled by the **Spoof** option, which is enabled by default. Half-NAT configurations are only a problem when a client is on the same subnet as the servers behind Equalizer, since the servers will try to respond directly back to the client -- which will not recognize the server connection as a response to it’s original request and so refuse the connection.

This "local client" problem is solved by *disabling* the **Spoof** option. When **Spoof** is disabled, Equalizer translates the source IP address in the request to one of Equalizer’s IP addresses before sending it on to the server. This is called *Source Network Address Translation*, or *SNAT* -- and this configuration is often called *Full-NAT*, since Equalizer is translating the client IP in packets from clients, as well as the server IP in packets from servers. In this case, servers will send responses to Equalizer’s IP address, so no special routing or gateway is needed on the server.

So, clusters with clients on a different subnet than the server pools behind it can have the spoof option enabled, while clusters with only local clients should have spoof disabled.

But what do you do if you expect client requests to come to the cluster from the local server subnet as well as other subnets?

In network configurations where Equalizer needs to be able to forward server responses to clients on the server subnet as well as other subnets for the same virtual cluster IP, the **Spoof** option can be selectively enabled or disabled by creating a Layer 7 match rule that looks for specific client IP addresses in incoming requests. When an incoming request’s source IP matches the rule, **Spoof** will be set as appropriate for that connection. This is commonly called *Selective SNAT*.

On Equalizer, implementing Selective SNAT using a Match Rule is the recommended method to allow local access to Layer 7 clusters with **Spoof** enabled; other alternatives include:

- adding static routes on all your server pools to clients on the server’s local subnet
- creating two clusters -- one on the non-server subnet with **spoof** enabled, and one on the server subnet with spoof disabled

Selective SNAT using a match rule is more easily implemented and maintained than either of the above methods, but can be configured only for Layer 7 clusters. If you require Selective SNAT with a Layer 4 cluster, you’ll need to use one of the above methods.

Selective SNAT Example

The procedure below shows you how to create a match rule that selectively disables the cluster **Spoof** option based on the client IP address of an incoming connection. It is assumed that the cluster for which the match rule is created has **Spoof** *enabled* on the cluster **Configuration** screen (tab), and that the cluster works properly for clients on subnets other than the subnet to which the server pools in the cluster are connected.

1. Right-click the name of the cluster for which you want to implement selective SNAT, and select **Add Match Rule.**
2. On the **Add New Match Rule** form:
   a. Type in a **Match Name** or accept the default.
   b. Select the **Next Match Rule** from the drop down list to place the new match rule in the desired order on the cluster.
   c. Click on **Commit.**

The new match rule is created and its **Configuration** Screen (tab) is opened.

3. Leave **any()** in the **expression** field.
4. In the **Expression Editor**:
   a. Drag and drop the **client_ip** function from the **Functions** pane to the **Expression Workbench.**
   b. Specify a simple IP address (e.g., "192.168.0.240"), or an IP address in Classless Inter- Domain Routing (CIDR) notation (e.g., "192.168.0.0/24") to specify an entire subnet in the **client_ip** function. Click on the **Continue** button when finished.

The **Expression** field should now contain the **client_ip** function with the **ip** argument you specified above.

5. Uncheck both the **Spoof** checkbox and the **Disable** checkbox on the **Configuration** Screen (tab).
6. Click on **Commit.**

Clients whose IP addresses are selected by the new match rule should now be able to connect successfully to the cluster IP. Right-click the name of the match rule in the left frame; the **Processed** counter in the popup menu should increase as clients are selected by the match rule. Select **Match Rule Plots** from the popup menu to display a history of the number of connections processed by the match rule.

Server Selection Based on Content Type Using Match Rules

In this example, assume a configuration that has dedicated one or more server pools to return only image files (*.gif*,*.jpg*, etc.), while the remainder of the server pools return all the other content for client requests.

We want to direct all requests for images to a particular server pool, and balance the remainder of requests across the other server pools in the cluster. The image server pool is connected to a common storage device that contains the images. The remaining server pools are all dedicated to serving particular content for different web sites. For this example, we assume that a cluster has already been defined.

We want to maintain persistent connections for the web site servers, assuming that some of the websites may need to maintain sessions for applications such as shopping carts, email, etc. Persistent connections are not necessary for the image servers, since they access the images from common storage and have no need to maintain client sessions, so there is no need to incur the performance impact of maintaining session information.

To do this, we'll create two match rules, as follows:

1. Log into the GUI using a login that has add/del access for the cluster.
2. In the navigation pane on the left, click the name of the Layer 7 cluster to which you want to add the rule. The cluster **Configuration** screen (tab) will appear on the right:
    a. Make sure that the **Once Only** checkbox is not checked; otherwise, uncheck it and click **Commit**.
    b. Make sure the **Persist** checkbox is not checked; otherwise, uncheck it and click **Commit**.

These steps are necessary because these flags, if enabled, cause only the first request in a connection to be evaluated. Since we want content to come from one server pool and images from another, we want the server pools that will have persistent connections to be chosen by the match rules.

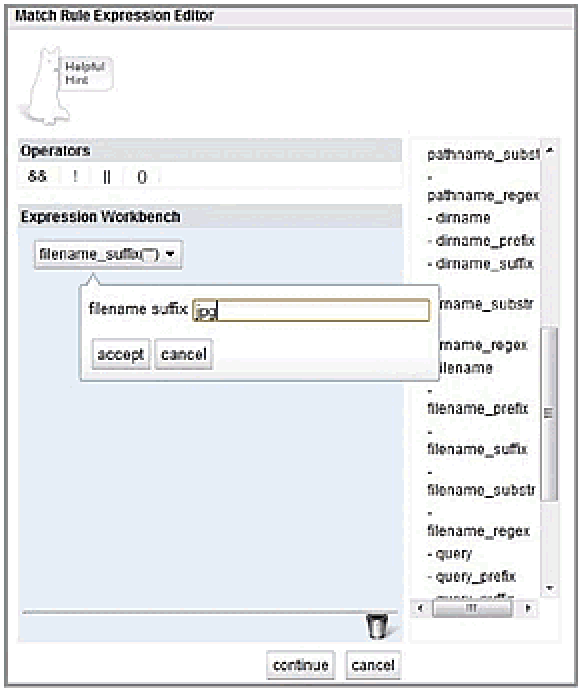
3. Right-click the cluster name in the left frame and select **Add Match Rule**. The **Add Match Rule** form appears:
    a. Type `images` into the **Match Name** text box and use the **Next Match Rule** drop-down list to specify the match rule order for this match rule on the selected cluster. Click on **Commit**. In this match rule, we'll construct an expression that will match all the filename extensions of the images to be served. These requests will go to the image servers.

The match rule is created, added to the object tree, and its **Configuration** tab is opened:

4. Click on the **Expression Editor** button:

   a. Drag and drop the **filename_suffix** function from the from **Functions** pane to the **ExpressionWorkbench.**

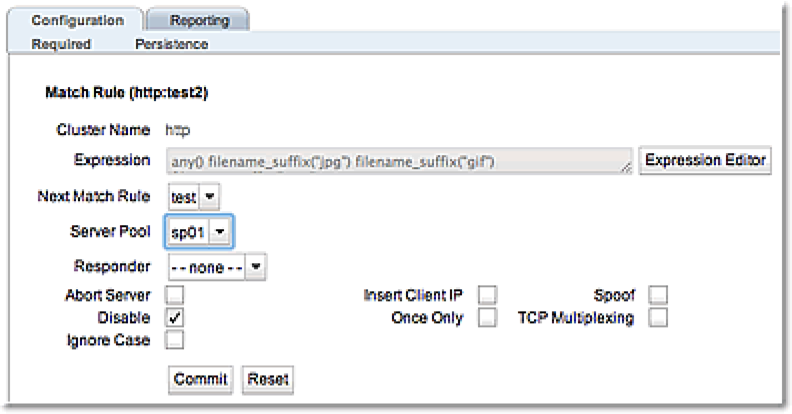
   b. Type "**jpg**" into the **filename suffix** text box.



   c. Select **continue.**

5. Repeat Step 4 for each of the other filename suffixes on our example servers -- **gif**, **bmp**, **tif** and **png.**

6. In our example, we want all the images to be served from **sp01.** On the **imagesConfiguration** screen (tab), select **sp01** from the **Server Pool** drop-down list. When you are done, the match expression should look like this:



Match Rule (http:test2)

Cluster Name	http
Expression	any() filename_suffix("jpg") filename_suffix("gif")
Next Match Rule	test
Server Pool	sp01
Responder	-- none --
Abort Server	☐
Disable	✓
Ignore Case	☐
Insert Client IP	☐
Once Only	☐
Spoof	☐
TCP Multiplexing	☐

7. Click on **Commit.**

The **images** rule we created selects all the requests for image files; now we need a rule to determine which servers will receive all the other requests. The Default rule is not sufficient, and in fact we don't want it to be reached, since it could send a request for content to one of the image servers. So, we'll create another rule with the same match expression as the Default [**any()**], but a restricted list of servers. This effectively *replaces* the Default match rule with one of our own.

8. In the left frame, right-click the name of the cluster and select **Add Match Rule.** The **Add Match Rule** screen appears:
    a. Type "**content"** into the **match name** text box and use the **Next Match Rule** drop-down list to specify the match rule order for this match rule on the selected cluster. Click on **Commit.**
    b. On the **Configuration** screen (tab) use the drop-down list to select the server pool in which all other content is to be sent.
    c. Select **Commit.**

The match rule is created, added to the object tree, and its **Configuration** Screen (tab) is opened:

9. Check the **Persist** check box. (Remember that in our example we're enabling **Persist** for the content servers, so that persistent sessions can be maintained by the applications that run on these servers.)
10. Select the **Commit** button to save your changes to the **Content** rule.

Cluster and Match Rule Statistics and Reporting (CLI and GUI)

The CLI display of Statistics can be seen by entering the following within the cluster or match rule context:

Sample of Layer 7 Cluster Statistical Display

Note - Refer to "Header Editing Reporting" on page 974 for a description of the Header Editing Statistics.

```
eqcli cl-Tes*> stats
                Current         60 sec          10 min          60 min
TOTALPRCSD      50891           34              37              27
TOTALRESPPRCSD  68535           60              63              43
TIMESPENT       45526           N/A             N/A             N/A
ACTIVECONX      242             0               256             186
BYTERCVD        20354896        N/A             N/A             N/A
BYTESEND        146733440       N/A             N/A             N/A
DROPNOSRVR      0               N/A             N/A             N/A
TOTALSTKY       0               N/A             N/A             N/A
CURRSTKY        0               0               0               0
REQPARSED       68535           N/A             N/A             N/A
REQFAILED       0               N/A             N/A             N/A
REQFAILHDR      0               N/A             N/A             N/A
RSPPARSED       0               N/A             N/A             N/A
RSPFAILED       0               N/A             N/A             N/A
RSPFAILHDR      0               N/A             N/A             N/A
CLNTTO          68111           N/A             N/A             N/A
SRVRTO          0               N/A             N/A             N/A
CONNTO          0               N/A             N/A             N/A
SELPERSIST      0               N/A             N/A             N/A
SPLICE          50891           N/A             N/A             N/A
CURCLNTWAITQ    0               N/A             N/A             N/A
CURCLNTWAITSRVR 0               N/A             N/A             N/A
CURCOMP         0               0               0               0
TOTALCOMP       0               N/A             N/A             N/A
INBYTECOMP      0               N/A             N/A             N/A
OUTBYTECOMP     0               N/A             N/A             N/A

Header Edit Statistic                                           Count
---------------------------------------------------------------------
Number of times client_request_edit() executed successfully:    2
Number of times client_request_edit() executed with errors:     0
Number of times server_response_edit() executed successfully:   2
Number of times server_response_edit() executed with errors:    0
eqcli cl-Tes*>
```

Sample of Layer 7 HTTP and HTTPS Match Rule Statistical Display

Note - Refer to "Header Editing Reporting" on page 974 for a description of the Header Editing Statistics.

```
eqcli cl-htt*-ma-Tes*> stats
                Current         60sec           10min           60min
TOTALPRCSD      6157678         4218            3028            2479

Header Edit Statistic                                           Count
----------------------------------------------------------------------
Number of times client_request_edit() executed successfully:    2
Number of times client_request_edit() executed with errors:     0
Number of times server_response_edit() executed successfully:   2
Number of times server_response_edit() executed with errors:    0
eqcli cl-htt*-ma-Tes*>
```

Sample of Layer 4 Cluster Statistical Display

```
eqcli cl-tes*> stats
                Current         60 sec          10 min          60 min
BYTERCVD        27781404        N/A             N/A             N/A
BYTESEND        138862736       N/A             N/A             N/A
DROPNOSRVR      0               N/A             N/A             N/A
TOTALSTKY       0               N/A             N/A             N/A
CURRSTKY        0               0               0               0

eqcli cl-tes*>
```

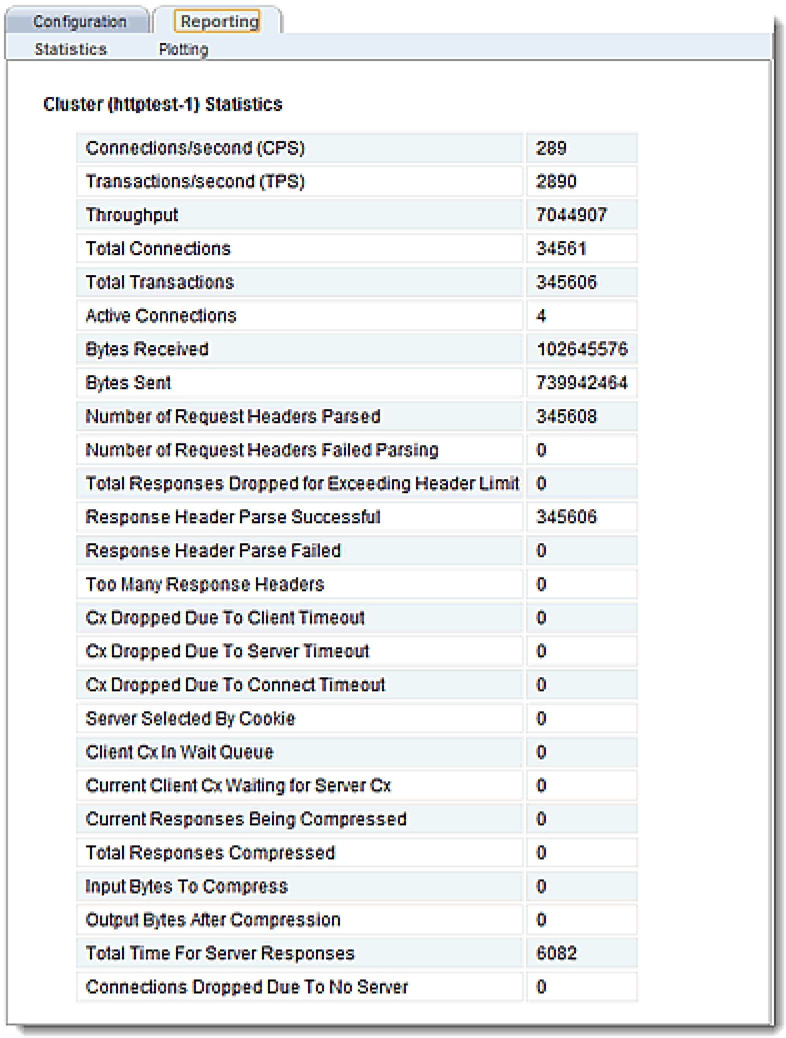
To view the GUI display:

1. Verify that you are logged into the GUI. (Refer to "Logging In" on page 238.)
2. Select the **Load Balance** configuration tab is it is not already selected.
3. Click on the arrow (▶) beside **Clusters** to expand the branch.
4. Select a cluster or responder Server on the left navigational pane and click on the **Reporting** tab to display statistics. The following is an example of the statistics displayed.

A Layer 4 statistical display is similar however it displays **Connections/second (CPS)**, **Throughput**, **Bytes Received**, **Bytes Sent**, **Total Sticky Records**, **Current Sticky Records**, and **Total Time for Server Responses.**

A Layer 7 Http and Https Match Rule statistical display is also similar however, it displays **Connections /second (CPS)** and **Total Connections.**

Sample Layer 7 Cluster GUI Statistical Displays



Cluster (httptest-1) Statistics

Connections/second (CPS)	289
Transactions/second (TPS)	2890
Throughput	7044907
Total Connections	34561
Total Transactions	345606
Active Connections	4
Bytes Received	102645576
Bytes Sent	739942464
Number of Request Headers Parsed	345608
Number of Request Headers Failed Parsing	0
Total Responses Dropped for Exceeding Header Limit	0
Response Header Parse Successful	345606
Response Header Parse Failed	0
Too Many Response Headers	0
Cx Dropped Due To Client Timeout	0
Cx Dropped Due To Server Timeout	0
Cx Dropped Due To Connect Timeout	0
Server Selected By Cookie	0
Client Cx In Wait Queue	0
Current Client Cx Waiting for Server Cx	0
Current Responses Being Compressed	0
Total Responses Compressed	0
Input Bytes To Compress	0
Output Bytes After Compression	0
Total Time For Server Responses	6082
Connections Dropped Due To No Server	0

The following are definitions for the statistical terms shown on both the CLI and GUI:

Layer 7 Cluster Statistic Definitions

CLI Term	GUI Term	Definition
TOTALPRCSD	Total Connections	Connections Processed.
TOTALRESPPRCSD	Total Transactions	The total responses processed.
TIMESPENT	Total Time For Server Responses	The total time spent on this object.
ACTIVECONX	Active Connections	Active Connections.
BYTERCVD	Bytes Received	Bytes received.
BYTESEND	Bytes Sent	Bytes transmitted.
REQPARSED	Number of Request Headers Parsed	This is the total number of times that an HTTP request header was parsed.
REQFAILED	Number of Request Headers	The number of request header failed
REQFAILHDR	Number of Request Headers Failed Parsing	The number of requests dropped for exceeding header limit.
RSPPARSED	Response Header Parse Successful	Total number of times HTTP response headers was parsed.
RSPFAILED	Response Header Parse failed	The number of response headers failed.
RSPFAILHDR	Too Many Response Headers	Too many response header.
CLNTTO	Cx dropped due to Client Timeout	Connections dropped due to client timeout.
SRVRTO	Cx Dropped due to Server Timeout	Connections dropped due to server timeout.
CONNTO	Cx Dropped Due to Connect Timeout	Connections dropped due to connect timeout
SELPERSIST	Server Selected By Cookie	The number of times the server was selected by a cookie.
SPLICE	Current Client Cx Waiting for Server Cx	Total client-server connections linked.
CURCLNTWAITQ	Client Cx in Wait Queue	The client connections in the queue.
CURCLNTWAITSRVR	Current Client Cx Waiting for Server Cx	The number of client connections waiting for server connections.
CURCOMP	Current Responses Being Compressed	The current responses being compressed.
TOTALCOMP	Total Responses Compressed	The total responses being compressed.
INBYTECOMP	Input Bytes to Compress	Input bytes to compress.

CLI Term	GUI Term	Definition
OUTBYTECOMP	Output Bytes after Compressions	Output byte after compression.

Layer 7 HTTP and HTTPS Match Rule Statistic Definitions

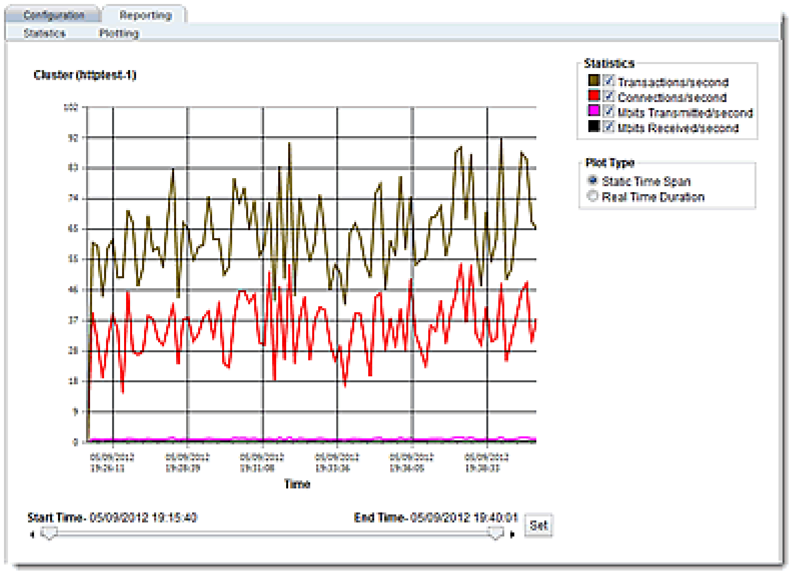
CLI Term	GUI Term	Definition
TOTALPRCSD	Connections/second (CSP)	Connections Processed.
N/A	Transactions/second (TPS)	The total responses processed.
N/A	Throughput	Throughput
N/A	Total Connections	Total connections.
N/A	Total Transactions	Total transactions.
N/A	Active Connections	Active connections.
N/A	Bytes Received	Bytes received.
N/A	Bytes Sent	Bytes transmitted.
N/A	Total Time For Server Responses	Total time for server responses
N/A	Connections Dropped Due To No Server	Connections dropped due to no server

Layer 4 Cluster Statistic Definitions

CLI Term	GUI Term	Definition
BYTERCVD	Bytes Received	Bytes received.
BYTESEND	Bytes Sent	Bytes transmitted.
N/A	Connections/second (CPS)	Connections per second.
DROPNOSRVR	N/A	Connections dropped due to no server.
TOTALSTKY	Total Sticky Records	Total sticky connections.
N/A	Throughput	Throughput.
CURRSTKY	Current Sticky Records	The current sticky record.
N/A	Total Time For Server Responses	Total time for server responses.

The following is an example of a graphical plot that can be displayed on the GUI. Select a Cluster or Match Rule on the left navigational pane and click on the **Reporting** tab and then **Plotting.** The following will be displayed:

Sample Layer 7 Cluster Graphical Plot



🟫 ✓ Transactions/second
🟥 ✓ Connections/second
🟪 ✓ Mbits Transmitted/second
⬛ ✓ Mbits Received/second

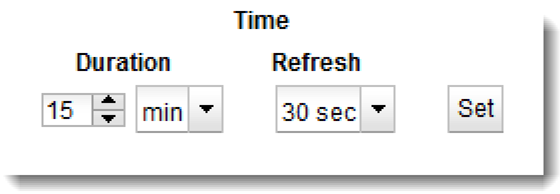
Plot Type: [● Static Time Span] ○ Real Time Duration

Start Time- 05/09/2012 19:15:40

End Time- 05/09/2012 19:40:01

The specific types of statistics that are displayed are determined by the selections on the **Statistics** pane on the upper right corner of the GUI.Make selections based on the data that you require.

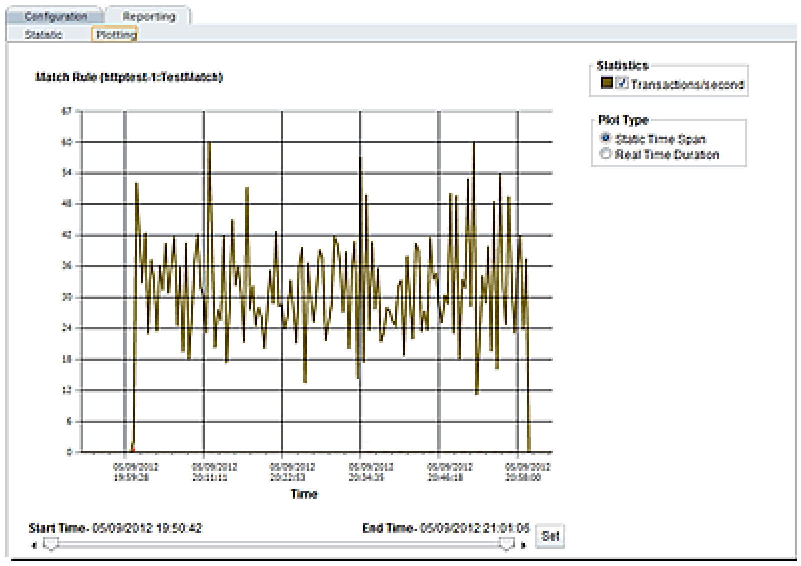
The **Plot Type** selection determines whether the display shown reflects a Static Time Span which is configured using the slider or whether a real time duration is display. If **Real Time Duration** is selected the slider controls will change to **Duration** and **Refresh** controls as shown below. In this case set the **Duration** of time in which you would like to review statistics and the **Refresh** rate desired.



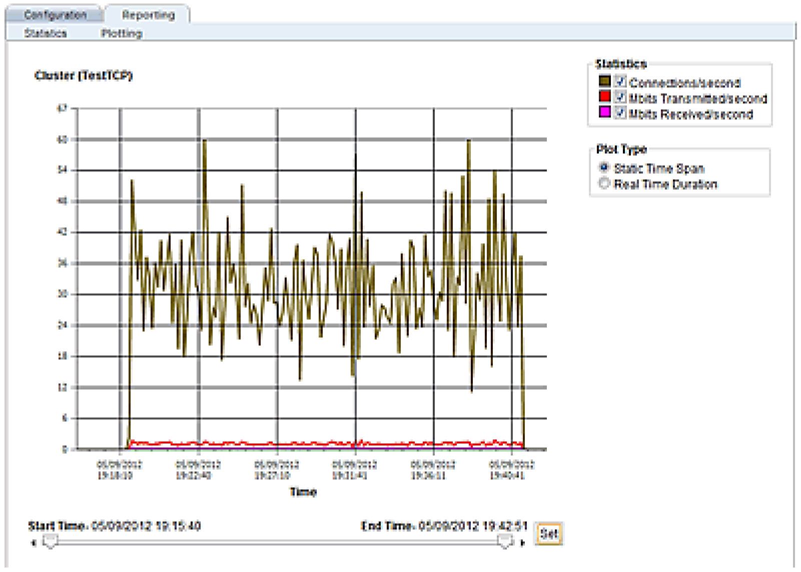
Duration: 15 min

Refresh: 30 sec

Sample Match Rule Graphical Plot

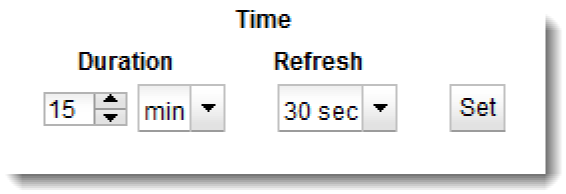


Sample Layer 4 Cluster Graphical Plot



The specific types of statistics that are displayed are determined by the selections on the **Statistics** pane on the upper right corner of the GUI.Make selections based on the data that you require.

The **Plot Type** selection determines whether the display shown reflects a Static Time Span which is configured using the slider or whether a real time duration is display. If **Real Time Duration** is selected the slider controls will change to **Duration** and **Refresh** controls as shown below. In this case set the **Duration** of time in which you would like to review statistics and the **Refresh** rate desired.



Chapter 13

Server Pools and Server Instances

Sections in this chapter include:

Managing Server Pools

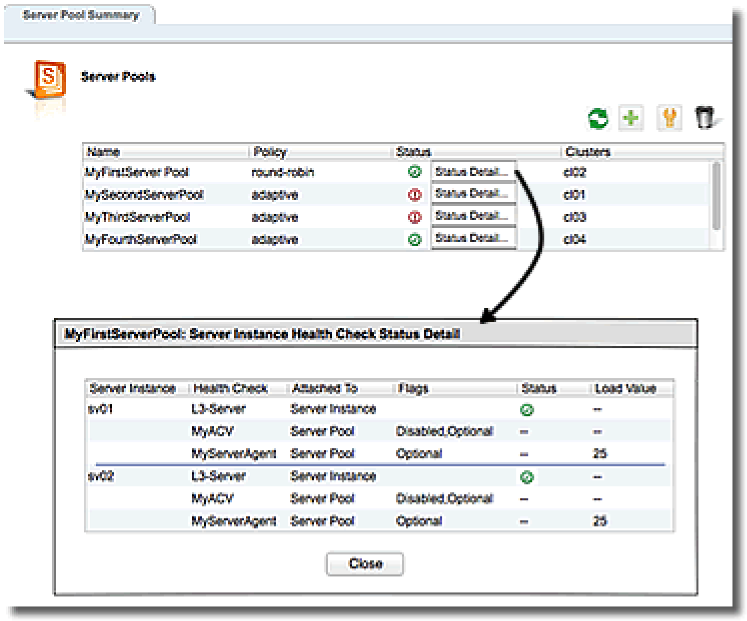
A server is attached to a cluster via a server pool. A server pool is a collection of server definitions, each of which has additional parameters assigned to it in the server pool -- these additional parameters are organized by the server’s name and are referred to as server instances within the server pool context. This allows you to associated a distinct set of server instance options (weight, flags, maximum number of connections), to multiple instances of the same real server in different server pools.

The following subsections describe Server Pool management using both the GUI and CLI.

Server Pool Summary

Server Pool Summary Screen on the GUI

The **Server Pool Summary** screen shown below will be displayed when you select the **Load Balance** configuration tab in the left navigational pane and then click on **Server Pools.** It displays the configured server pools, Load Balancing Policy, Status and the Clusters associated with them.



Name	Policy	Status		Clusters
MyFirstServer Pool	round-robin	✓	Status Detail...	cl02
MySecondServerPool	adaptive	!	Status Detail...	cl01
MyThirdServerPool	adaptive	!	Status Detail...	cl03
MyFourthServerPool	adaptive	✓	Status Detail...	cl04

MyFirstServerPool: Server Instance Health Check Status Detail

Server Instance	Health Check	Attached To	Flags	Status	Load Value
sv01	L3-Server	Server Instance		✓	--
	MyACV	Server Pool	Disabled,Optional	--	--
	MyServerAgent	Server Pool	Optional	--	25
sv02	L3-Server	Server Instance		✓	--
	MyACV	Server Pool	Disabled,Optional	--	--
	MyServerAgent	Server Pool	Optional	--	25

Double click on any of the **Server Pools** in the list to and the screen will expand to display the **Server Pool Settings** configuration screen.

Status Indicators

These icons provide status of the server instances in the server pools.

✓ This icon indicates that the server instances in the server pool are up and running.

! This icon indicates that one or more of the server instances in the server pool are down.

Server Pool Summary using the CLI

To view server pool parameters and properties using the GUI use the following example, replacing **`srvpool1`** with the name of the server pool that you would like to view:

```
eqcli > show srvpool srvpool1
Server Pool Name                 : srvpool1
Load balancing Policy            : round-robin
Load balancing Responsiveness    : 3
Flags                            :
Custom lb: healthcheck weight    : 33
Custom lb: connection weight     : 33
Custom lb: delay weight          : 33

eqcli >
```

Configuring Server Pool Load-Balancing Options

Configure load balancing policy and response settings for each server pool independently. Multiple clusters do not need to use the same load balancing configuration even if the same physical server machines host them. For example, if one cluster on port 80 handles HTML traffic and one on port 8000 serves images, you can configure different load balancing policies for each server pool.

When you use adaptive load balancing (that is, you have *not* set the cluster's load balancing policy to round robin or static weight), you can adjust Equalizer to optimize performance.

Equalizer's Load Balancing Policies

Equalizer supports the following load balancing policies, each of which is associated with a particular algorithm that Equalizer uses to determine how to distribute requests among the servers in the server pool:

- **Round-robin** load balancing - distributes requests equally on the server pool in the cluster. Equalizer dispatches the first incoming request to the first server, the second to the second server, and so on. When Equalizer reaches the last server, it repeats the cycle. If a server in the cluster is down, Equalizer does not send requests to that server. This is the default method.

  The round robin method does not support Equalizer's adaptive load balancing feature; so, Equalizer ignores the servers' initial weights and does not attempt to dynamically adjust server weights based on server performance.
- **Static** load balancing - distributes requests among the servers depending on their assigned initial weights. A server with a higher initial weight gets a higher percentage of the incoming requests. Think of this method as a *weighted round robin* implementation. Static weight load balancing does not support Equalizer's adaptive load balancing feature; Equalizer does not dynamically adjust server weights based on server performance.
- **Adaptive** load balancing - distributes the load according to the following performance indicators for each server.
- **Response** is the length of time for the server to begin sending reply packets after Equalizer sends a request.
- **Least Cxns** (least connections) load balancing - dispatches the highest percentage of requests to the server with the least number of active connections. In the same way as Fastest Response, Equalizer tries to avoid overloading the server so it checks the server's response time and server agent value. Least Connections optimizes the balance of connections to servers in the cluster.
- **Server Agent** load balancing - dispatches the highest percentage of requests to the server with the lowest server agent value. In a similar way to Fastest Response, Equalizer tries to avoid overloading the server by checking the number of connections and response time. This method only works if the server agents are running on every server instance in the server pool.
- **Custom** load balancing - If custom is selected, you can adjust the load balancing policy parameters:

- **Connections** - The relative influence on the policy of the number of active connections currently open to a server
- **Response Time**load balancing - dispatches the highest percentage of requests to the server with the shortest response time. Equalizer does this carefully: if Equalizer sends too many requests to a server, the result can be an overloaded server with slower response time. The fastest response policy optimizes the cluster-wide response time. The fastest response policy also checks the number of active connections and server agent values (if configured); but both of these have less of an influence than they do under the adaptive load balancing policy. For example, if a server's active connection count and server agent values are high, Equalizer might not dispatch new requests to that server even if that server's response time is the fastest in the cluster.
- **Health Checks** - This is the relative weight for the aggregate health check response value in load balancing calculations.
- **Responsiveness** - Determines how aggressively server dynamic weights are optimized as a cluster load changes.

Aggressive Load Balancing

After you fine-tune the initial weights of each server in the cluster, you might discover that Equalizer is not adjusting the dynamic weights of the servers at all: the dynamic weights are very stable, even under a heavy load. In this case, you might want to set the cluster's load balancing response parameter to *fast*. Then Equalizer tries to optimize the performance of your servers more aggressively; this should improve the overall cluster performance. For more information about setting server weights, see "Adjusting a Server's Initial Weight" on page 534.

Dynamic Weight Oscillations

If you notice a particular server's dynamic weight oscillates (for example, the dynamic weight varies from far below 100 to far above 100 and back again), you might benefit by choosing *slow* response for the cluster. You should also investigate the reason for this behavior; it is possible that the server application is behaving erratically.

Using Active Content Verification (ACV)

Active Content Verification (ACV) is a mechanism for checking the validity of a server. When you enable ACV for a server pool, Equalizer requests data from each server instance in the server pool and verifies that the returned data contains a character string that indicates that the data is valid. You can use ACV with most network services that support a text-based request/response protocol, such as HTTP. Note, however, that you cannot use ACV with Layer 4 UDP clusters.

ACV checking is performed as part of the high-level TCP probes that Equalizer sends to every server by default. To enable ACV, you specify an **ACV response** string for a server pool. Equalizer which will then search for the **ACV response** string in the first 1024 characters of the server's response to the high-level TCP probes. If the ACV response string is not found, the server is marked down. An ACV probe can be specified if the service running on the server's **probe port** requires input in order to respond.

How ACV works is best explained using a simple example. The HTTP protocol enables you to establish a connection to a server, request a file, and read the result.

`> telnet www.myserver.com 80 >>>>`	User requests connection to server.
`Connected to www.myserver.com >>>>`	Telnet indicates connection is established.
`> GET /index.html >>>>`	User sends request for HTML page.
`<HTML> >>>>>`	Server responds with requested page.
`<TITLE>Welcome to our Home Page </TITLE>`	
`</HTML>`	
`Connection closed by foreign host >>>>`	Telnet indicates server connection closed.

Equalizer can perform the same exchange automatically and verify the server's response by checking the returned data against an expected result.

Specifying an *ACV probe string* and an *ACV response string* basically automates the above exchange. Equalizer uses the probe string to request data from each server. To verify the server's content, Equalizer searches the returned data for the response string. For example, you can use "`GET /index.html`" as the *ACV probe string* and you can set the response string to some text, such as "`Welcome`" appears on the home page.

Similarly, if you have a Web server with a PHP application that accesses a database, you can use ACV to ensure that all the components of the application are working. You could set up a PHP page called **test.php** that accesses the database and returns a page containing "ALL OK" if there are no problems.

Then you would enter and ACV Query and an ACV Response String using either the CLI or GUI. :

If the page that is returned contains the correct response string in the first 1000 characters, including headers) the server is marked "up"; if "ALL OK" were not present, the server is marked down.

The response string should be text that appears only in a valid response. This string is case-sensitive. An example of a poorly chosen string would be “HTML”, since most web servers automatically generate error pages that contain valid HTML.

For more information on probing, see "Configuring ACV Health Checks" on page 567.

Adding and Configuring Server Pools

Add and configure Server Pools using the GUI or CLI as follows:

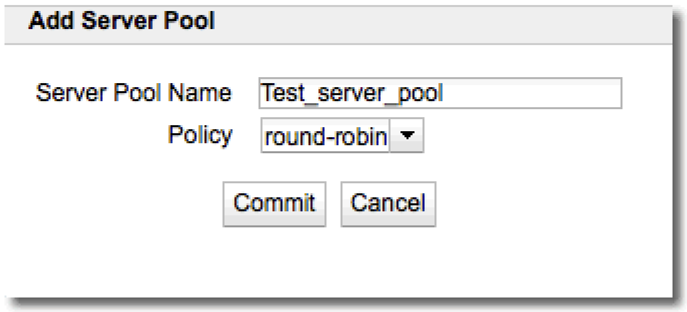
Server Pool Parameters

GUI Parameter (CLI Parameter)	Description
Disable **(`disable`)**	Selecting the Disable option will disable the Server Pool
Responsiveness **(`respv`)**	The **Responsiveness** setting controls how aggressively Equalizer adjusts the servers' dynamic weights. Equalizer provides five response settings: **Slowest**, **Slow**, **Medium**, **Fast**, and **Fastest**. The response setting affects the dynamic weight spread, weight spread coefficient, and optimization threshold that Equalizer uses when it performs adaptive load balancing: **Dynamic Weight Spread** indicates how far a server's dynamic weight can vary (or *spread*) from its initial weight. **Weight Spread Coefficient** regulates the speed of change to a server's dynamic weight. The weight spread coefficient causes dynamic weight changes to happen more slowly as the difference between the dynamic weight and the initial weight increases. **Optimization Threshold** controls how frequently Equalizer adjusts dynamic weights. If Equalizer adjusts server weights too aggressively, oscillations in server weights can occur and cluster-wide performance can suffer. On the other hand, if Equalizer does not adjust weights often enough, server overloads might not be compensated for quickly enough and cluster-wide performance can suffer.
For Custom Load Balancing Policy:	
Connections **(`custom_actconn`)**	The active connections weight for the custom load balancing policy.
Response Time **(`custom_delay`)**	The delay weight for the custom load balancing policy.
Health Checks **(`custom_hc`)**	The health check weight for the custom load balancing policy.

Adding and Configuring a Server Pool (GUI)

To add and configure a server pool using the GUI proceed with the following:

1. Log in to the GUI as described in "Logging In" on page 238.
2. Select the **Load Balance** configuration tab is it is not already selected.
3. Right click on Server Pools on the object tree and select **Add Server Pool**. The following will be displayed.

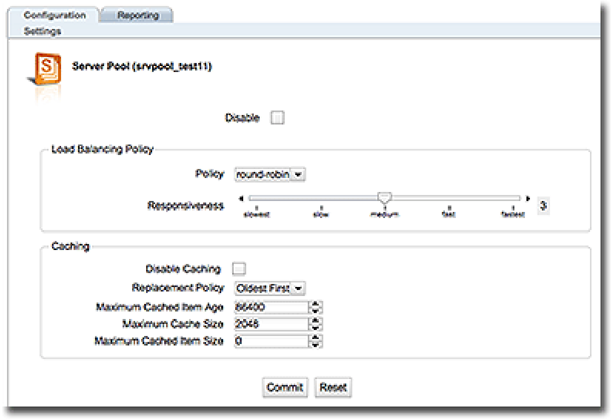


Add Server Pool

Server Pool Name: Test_server_pool

Policy: round-robin

4. Add a name for the server pool in the **Server Pool Name** field and select the load balancing policy using the **Policy** drop-down list. Click on **Commit** to save the Server Pool. It will appear on the **Server Pool** tree on the left navigational pane and the Server Pool **Configuration > Settings** screen will appear on the right. Note that if you select a **Custom** Policy you will have the option of configuring the Custom Load Balancing policies discussed in "Configuring Server Pool Load-Balancing Options" on page 479.



Server Pool (srvpool_test11)

Disable: ☐

Load Balancing Policy

Policy: round-robin

Responsiveness: 3

Caching

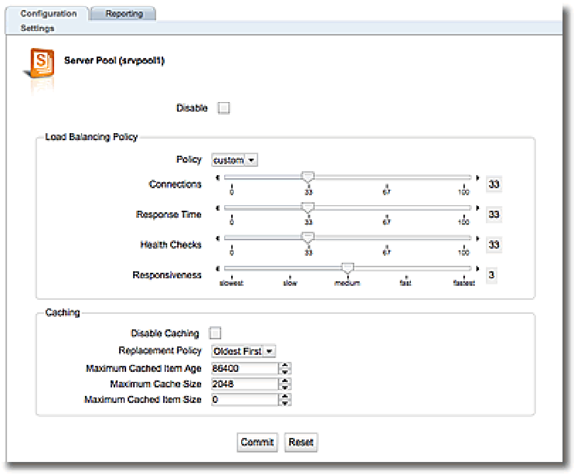
Disable Caching: ☐

Replacement Policy: Oldest First

Maximum Cached Item Age: 86400

Maximum Cache Size: 2048

Maximum Cached Item Size: 0



Server Pool (srvpool1)

Disable: ☐

Load Balancing Policy

Policy: custom

Connections: 33

Response Time: 33

Health Checks: 33

Responsiveness: 3

Caching

Disable Caching: ☐

Replacement Policy: Oldest First

Maximum Cached Item Age: 86400

Maximum Cache Size: 2048

Maximum Cached Item Size: 0

Server pool caching is disabled, by default. When the Disable Caching check box is un-checked to enable caching, additional parameter configuration is required. Refer to "Using the CLI and GUI to Configure Caching" on page 509 for descriptions of the caching function and configuration.

5. Configure Server Pool parameters using the table above.
6. Click on **Commit** to save the configuration.

Adding and Configuring a Server Pool (CLI)

To add and configure a server pool using the CLI proceed with the following:

1. Log in to the CLI as described in "Starting the CLI" on page 142 .
2. Enter the new server pool details in the following format at the command line to create a server pool:

```
eqcli > srvpool spname req_cmds
```

3. Use the load balancing options as described above in "Configuring Server Pool Load-Balancing Options" on page 479 and the "Server Pool, Server Instance, and Caching Commands" on page 209 to configure the other server pool parameters.

Adding Server Instances

A server pool is a collection of server definitions, each of which has additional parameters assigned to it in the server pool -- these additional parameters are organized by the server's name and are referred to as *server instances*.

Parameters

GUI Parameter (CLI Parameter)	Description
Initial Weight **(weight)**	An number between 0 and 200 that indicates a server's processing power relative to the other servers in a cluster. The default is 100. A value of 0 disables the server (no traffic will be routed to the server). For information about selecting an appropriate initial weight, refer to "Adjusting a Server's Initial Weight" on page 534.
Maximum Connections **(maxconn)**	Sets the maximum number of permitted open connections for the server. Once this limit is reached, no more traffic is routed to the server until the number of open connections falls below this limit. This limit is set by default to 0, which means that there is no maximum connections limit on the server.
Hot Spare **(hot_spare)**	Enable the hot spare check box if you plan to use this server as a backup server, in case the other server instances in a server pool on the cluster fail. Checking hot spare forces Equalizer to direct incoming connections to this server only if all the other servers in the cluster are down. You should only configure one server in a cluster as a hot spare. For example, you might configure a server as a hot spare if you are using licensed software on your servers and the license allows you to run the software only on one node at a time. In this situation, you could configure the software on two servers in the cluster and then configure one of those servers as a hot spare. Equalizer will use the second server only if the first goes down, enabling you to make your application available without violating the licensing terms or having to purchase two software licenses.
Override Persistence **(persist_override)**	Disables persistence for the server when the persist flag (Layer 7 cluster) or a non-zero sticky time (Layer 4 cluster) is set on a cluster. For a Layer 7 cluster, this means that a cookie will not be inserted into the response header when returned to the client. No sticky record is set for a Layer 4 cluster. This flag is usually used to disable persistence for a hot spare.
Quiesce **(quiesce)**	When enabled, Equalizer avoids sending new requests to the server. This is usually used in preparation for shutting down an HTTP or HTTPS server, and is sometimes also called "server draining". Refer to "Shutting Down a Server Gracefully" on page 539.

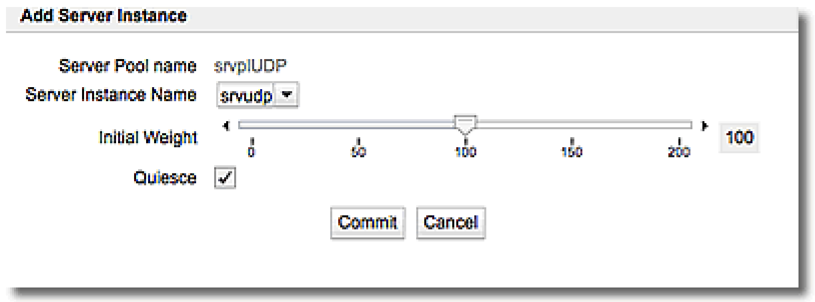
GUI Parameter (CLI Parameter)	Description
Strict Max Cx **(strict_maxconn)**	This flag allows you to customize the behavior of the max connections parameter (see above). When **Strict Max Cx** is enabled (the default), the max connections parameter is interpreted as a strict maximum and is never overridden. If a client attempts to connect to a server that has a number of connections equal to the max connections setting, then the connection is refused. When **Strict Max Cx** is disabled, the max connections setting will be overridden in any of the following circumstances: A client attempts to connect to a server with the hot spare flag enabled - this allows hot spares to service more than the max connections setting of connections. A client attempting to connect to a Layer 7 cluster has a persistence cookie and the server identified in the cookie has already reached its max connections limit. A client attempting to connect to a Layer 4 cluster has an existing sticky persistence connection to a server and that server has already reached its max connections limit.
Initial Weight **(weight)**	An number between 0 and 200 that indicates a server's processing power relative to the other servers in a cluster. The default is 100. A value of 0 disables the server (no traffic will be routed to the server). For information about selecting an appropriate initial weight, refer to "Adjusting a Server's Initial Weight" on page 534.
Maximum Connections **(maxconn)**	Sets the maximum number of permitted open connections for the server. Once this limit is reached, no more traffic is routed to the server until the number of open connections falls below this limit. This limit is set by default to 0, which means that there is no maximum connections limit on the server.
Hot Spare **(hot_spare)**	Enable the hot spare check box if you plan to use this server as a backup server, in case the other server instances in a server pool on the cluster fail. Checking hot spare forces Equalizer to direct incoming connections to this server only if all the other servers in the cluster are down. You should only configure one server in a cluster as a hot spare. For example, you might configure a server as a hot spare if you are using licensed software on your servers and the license allows you to run the software only on one node at a time. In this situation, you could configure the software on two servers in the cluster and then configure one of those servers as a hot spare. Equalizer will use the second server only if the first goes down, enabling you to make your application available without violating the licensing terms or having to purchase two software licenses.
Override Persistence **(persist_override)**	Disables persistence for the server when the persist flag (Layer 7 cluster) or a non-zero sticky time (Layer 4 cluster) is set on a cluster. For a Layer 7 cluster, this means that a cookie will not be inserted into the response header when returned to the client. No sticky record is set for a Layer 4 cluster. This flag is usually used to disable persistence for a hot spare.

Quiesce **(quiesce)**	When enabled, Equalizer avoids sending new requests to the server. This is usually used in preparation for shutting down an HTTP or HTTPS server, and is sometimes also called "server draining". Refer to "Shutting Down a Server Gracefully" on page 539.
Strict Max Cx **(strict_maxconn)**	This flag allows you to customize the behavior of the max connections parameter (see above). When **Strict Max Cx** is enabled (the default), the max connections parameter is interpreted as a strict maximum and is never overridden. If a client attempts to connect to a server that has a number of connections equal to the max connections setting, then the connection is refused. When **Strict Max Cx** is disabled, the max connections setting will be overridden in any of the following circumstances: A client attempts to connect to a server with the hot spare flag enabled - this allows hot spares to service more than the max connections setting of connections. A client attempting to connect to a Layer 7 cluster has a persistence cookie and the server identified in the cookie has already reached its max connections limit. A client attempting to connect to a Layer 4 cluster has an existing sticky persistence connection to a server and that server has already reached its max connections limit.

Adding Server Instances(GUI)

Add server instances as follows:

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Load Balance** configuration tab if it is not already selected.
3. Click on the arrow (▶) beside **Server Pools** to expand the branch.
4. Click on the arrow (▶) beside **Servers** to expand the branch.
5. Select one of the servers in the list of **Servers** on the tree on left navigational pane. While holding your mouse key down, drag and drop the server into the desired server pool on the server pool branch of the tree. The figure below will be displayed.



Add Server Instance

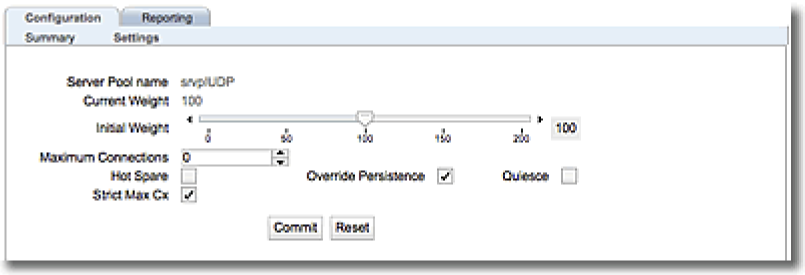
Server Pool name srvplUDP

Server Instance Name srvudp

Initial Weight 100

Quiesce ✓

6. Select a Server Instance to use from the **Server Instance Name** drop down list.
7. Select the **Initial Weight** for the server instance using the slider control.
8. Disable the **Quiesce** check box, if desired. By default, this option is enabled and causes server pool to ignore this server for new connections. If the server is already configured and ready to accept connections, disable this option. Otherwise, leave this option enabled and disable it once the server is ready. Click **Commit**. The figure below will be displayed.



Server Pool name: srvp/UDP
Current Weight: 100
Initial Weight: 100 (slider 0, 50, 100, 150, 200)
Maximum Connections: 0
Hot Spare: ☐
Override Persistence: ✓
Quiesce: ☐
Strict Max Cx: ✓

9. Configure the server instance using the parameters in the table above.

Note - For servers in Layer 7 HTTPS clusters, set the probe port to something other than 443, since Equalizer communicates with the servers via HTTP. In many configurations, it is set to the server port. The server agent port, set on the cluster, remains a separate port that is used only for server agent communication.)

8. Repeat step 5 to add additional server instances to a server pool.
9. Click on **Commit** to save the Server Pool configuration.

Adding Server Instances (CLI)

Server instance specific commands can be applied to multiple server instances by entering a comma-separated list of server instance names on the command line. For example, to set the weight to 125 on three server instances (sv01, sv02, sv03) in server pool sp01, you could enter a command like this:

```
eqcli > srvpool sp01 si sv01,sv02,sv03 weight 125
```

You can also change to an aggregate context that applies to multiple server instances, that allows you to display and modify the parameters for all the server instances. For example, you could change to an aggregate context for the three server instances in the previous example above using a command like the following:

```
eqcli > srvpool sp01 si sv01,sv02,sv03
eqcli sp-sp01-si-sv0*>
```

The CLI is now in the aggregate server instance context "**sv01,sv02,sv03**" -- only the first three characters of which are displayed in the command line. To see the entire context name, use the **context** command:

```
eqcli sp-sp01-si-sv0*> context
```

The context is "sv01,sv02,sv03".

```
eqcli sp-sp01-si-sv0*>
```

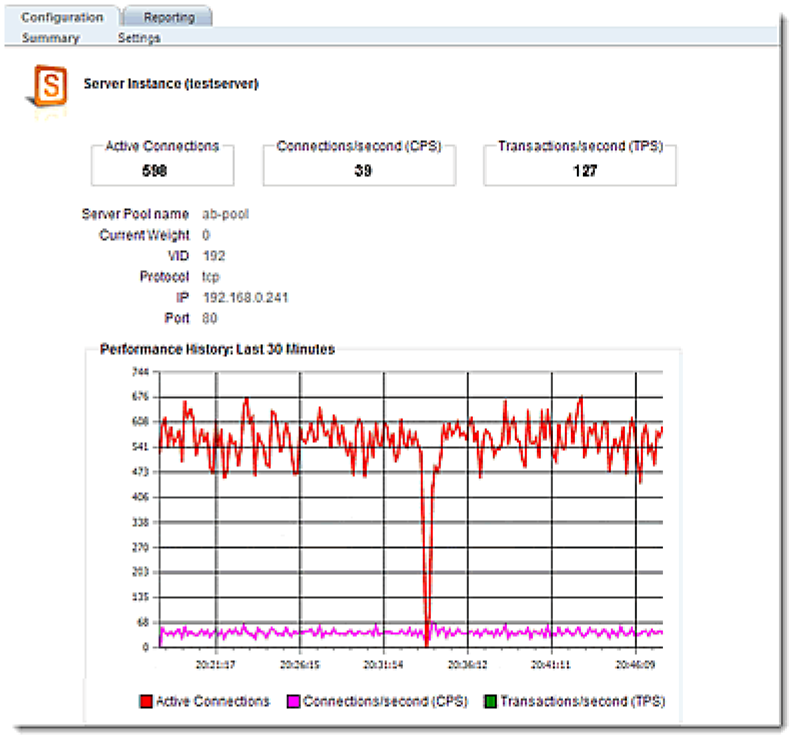
In an aggregate server instance context, the **show** command will display the configuration of all the server instances in the context.

Configure the server instance using the parameters in the table above

Note - For servers in Layer 7 HTTPS clusters, set the probe port to something other than 443, since Equalizer communicates with the servers via HTTP. In many configurations, it is set to the server port. The server agent port, set on the cluster, remains a separate port that is used only for server agent communication.)

Server Instance Summary Screen

The **Server Instance Summary Screen** is displayed when a server instance is selected from the left navigational pane. It displays server instance details such as **Active Connections**, **Connections/second** and **Transactions per second** as well as server pool configuration parameters and a graphical representation of performance history from the last 30 minutes.



Active Connections	Connections/second (CPS)	Transactions/second (TPS)
598	39	127

Server Pool name	ab-pool
Current Weight	0
VID	192
Protocol	tcp
IP	192.168.0.241
Port	80

Associating a Server Pool with a Cluster

Server Pools can be associated with clusters using either the GUI or the CLI>

Using the GUI

To associate a server pool with a cluster proceed with the following:

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Load Balance** configuration tab if it is not already selected.
3. Click on the arrow (▶)beside **Clusters** to expand the branch.
4. Select a **Cluster** and the **Configuration Required** screen will be displayed.
5. Select a server pool from the **Server Pool** drop down list.
6. Refer to "Overview" on page 324and make any additional changes to the cluster configuration if necessary.
7. Click on **Commit** to save new server pool association with the cluster.

Using the CLI

To associate a server pool with a cluster proceed with the following:

1. Access the CLI as described in "Starting the CLI" on page 142.
2. Use the following format to enter the cluster context.

```
eqcli> cluster clname
```

3. In the cluster context enter details in the following format:

```
eqcli cl-clname> srvpool spname
```

Deleting Server Pools

Server Pools can be deleted using either the GUI or the CLI.

Delete Using the GUI

Proceed with the following to remove a server pool using the GUI:

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238
2. Click on the **Load Balance** configuration tab if it is not already selected.
3. Click on the arrow (▶) beside **Server Pools** to expand the branch.
4. Right click on the server pool to be deleted from the **Server Pool** branch of the tree on the left navigational pane and select **Delete Server Pool.**
5. Click on **Confirm** when prompted on the **Delete Server Pool** dialogue form.

Delete Using the CLI

Proceed with the following to remove a server pool using the CLI:

1. Access eqcli as described in "Starting the CLI" on page 142 .
2. Use the following format to enter the cluster context.

```
eqcli> cluster clname
```

3. In the cluster context enter details in the following format:

```
eqcli cl-clname> no   srvpool spname
```

Server Pool and Server Instance Reporting

The CLI display of Statistics can be seen by entering the following within the server pool context:

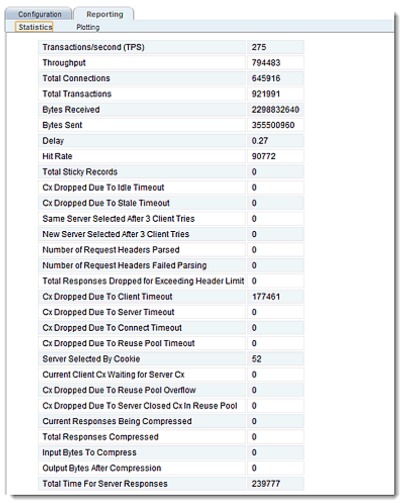
Sample Server Pool or Server Instance Statistical Display

```
eqcli sp-srv*> stats
                    Current     60 sec      10 min      60 min
TOTALPRCSD          391468      94          97          78
TOTALRESPPRCSD      562924      125         128         127
TIMESPENT           159054      N/A         N/A         N/A
ACTIVECONX          747         741         766         535
BYTERCVD            1397537644  N/A         N/A         N/A
BYTESEND            215641616   N/A         N/A         N/A
TOTALSTKY           0           N/A         N/A         N/A
IDLECONXDROPED      0           N/A         N/A         N/A
STALECONXDROPED     0           N/A         N/A         N/A
FAILTHRICE          0           N/A         N/A         N/A
NEWFAILTHRICE       0           N/A         N/A         N/A
RSPPARSED           483043      N/A         N/A         N/A
RSPFAILED           0           N/A         N/A         N/A
RSPFAILHDR          0           N/A         N/A         N/A
CLNTTO              104511      N/A         N/A         N/A
SRVRTO              0           N/A         N/A         N/A
CONNTO              0           N/A         N/A         N/A
SELPERSIST          0           N/A         N/A         N/A
SPLICE              311587      N/A         N/A         N/A
CURCLNTWAITQ        0           N/A         N/A         N/A
REUSEOF             0           N/A         N/A         N/A
REUSESRVR           0           N/A         N/A         N/A
REUSETO             0           N/A         N/A         N/A
CURCOMP             0           0           0           0
TOTALCOMP           0           N/A         N/A         N/A
INBYTECOMP          0           N/A         N/A         N/A
OUTBYTECOMP         0           N/A         N/A         N/A
eqcli sp-srv*>
```

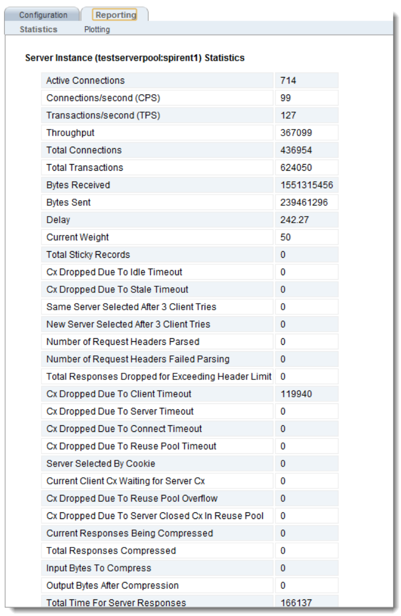
To view the GUI display:

1. Verify that you are logged into the GUI. (Refer to "Logging In" on page 238.)
2. Select the **Load Balance** configuration tab on the left navigational pane if it is not already selected.
3. Click on the arrow (▶) beside **Server Pools**to expand the branch.
4. Select a Server Pool or Server Instance Server on the branch and click on the **Reporting** tab to display statistics. The following is an example of the statistics displayed.

Sample Server Pool and Server Instance GUI Statistical Display



Transactions/second (TPS)	275
Throughput	794483
Total Connections	645916
Total Transactions	921991
Bytes Received	2298832640
Bytes Sent	355500960
Delay	0.27
Hit Rate	90772
Total Sticky Records	0
Cx Dropped Due To Idle Timeout	0
Cx Dropped Due To Stale Timeout	0
Same Server Selected After 3 Client Tries	0
New Server Selected After 3 Client Tries	0
Number of Request Headers Parsed	0
Number of Request Headers Failed Parsing	0
Total Responses Dropped for Exceeding Header Limit	0
Cx Dropped Due To Client Timeout	177461
Cx Dropped Due To Server Timeout	0
Cx Dropped Due To Connect Timeout	0
Cx Dropped Due To Reuse Pool Timeout	0
Server Selected By Cookie	52
Current Client Cx Waiting for Server Cx	0
Cx Dropped Due To Reuse Pool Overflow	0
Cx Dropped Due To Server Closed Cx In Reuse Pool	0
Current Responses Being Compressed	0
Total Responses Compressed	0
Input Bytes To Compress	0
Output Bytes After Compression	0
Total Time For Server Responses	239777



Server Instance (testserverpool:spirent1) Statistics

Active Connections	714
Connections/second (CPS)	99
Transactions/second (TPS)	127
Throughput	367099
Total Connections	436954
Total Transactions	624050
Bytes Received	1551315456
Bytes Sent	239461296
Delay	242.27
Current Weight	50
Total Sticky Records	0
Cx Dropped Due To Idle Timeout	0
Cx Dropped Due To Stale Timeout	0
Same Server Selected After 3 Client Tries	0
New Server Selected After 3 Client Tries	0
Number of Request Headers Parsed	0
Number of Request Headers Failed Parsing	0
Total Responses Dropped for Exceeding Header Limit	0
Cx Dropped Due To Client Timeout	119940
Cx Dropped Due To Server Timeout	0
Cx Dropped Due To Connect Timeout	0
Cx Dropped Due To Reuse Pool Timeout	0
Server Selected By Cookie	0
Current Client Cx Waiting for Server Cx	0
Cx Dropped Due To Reuse Pool Overflow	0
Cx Dropped Due To Server Closed Cx In Reuse Pool	0
Current Responses Being Compressed	0
Total Responses Compressed	0
Input Bytes To Compress	0
Output Bytes After Compression	0
Total Time For Server Responses	166137

The following are definitions for the statistical terms shown on both the CLI and GUI:

Server Pool Statistic Definitions

CLI Term	GUI Term	Definition
Total connections processed	Total Connections	Connections Processed.
Total response processed	Total Transactions	Responses Processed.
Total time taken for server to respond	Total Time For Server Responses	Total Time For Server Responses
Current Active Connections	Active Connections	Active Connections.
No of Times Server Selected By Sticky	Total Sticky Records	Total Sticky Records.
Total New Server Selected after 3 Retries	New Server Selected After 3 Client Tries	New Server Selected After 3 Client Tries.
Total Same Server Selected after 3 Retries	Same Server Selected After 3 Client Tries	Same Server Selected After 3 Client Tries.
Total Connections Dropped for Stale Timeout	Cx Dropped Due To Stale Timeout	Connections dropped due to stale timeout.
Total Connections Dropped for Idle Timeout	Cx Dropped Due To Idle Timeout	Total Connections Dropped for Idle Timeout.
Number of Request Headers Failed Parsing	Number of Request Headers Failed Parsing	Number of Request Headers Failed Parsing.
Total Responses Failed Header Parsing	Number of Request Headers Failed Parsing	Number of Request Headers Failed Parsing.
Total Responses Dropped for Exceeding Header Limit	Total Responses Dropped for Exceeding Header Limit	Total Responses Dropped for Exceeding Header Limit.
No of Times Server Selected By Cookie	Server Selected By Cookie	No of Times Server Selected By Cookie.
Cx Dropped Due To Client Timeout	Cx Dropped Due To Client Timeout	Connections dropped due to client timeout.
Cx Dropped Due To Server Timeout	Cx Dropped Due To Server Timeout	Connections dropped due to server timeout
Cx Dropped Due To Connect Timeout	Cx Dropped Due To Connect Timeout	Connections dropped due tto connect timeout.
Current Connections in TCP MUX Reuse Pool	Cx Dropped Due To Reuse Pool Timeout	Connections dropped due to reuse pool timeout.
Total Client-Server Connections Linked	Current Client Cx Waiting for Server Cx	Total client-server connections linked
Total Connections Timed Out in TCP MUX Reuse Pool	Cx Dropped Due To Reuse Pool Timeout	Total connections timed out in TCP MUX reuse pool.

CLI Term	GUI Term	Definition
Total Connections Terminated for TCP MUX Reuse Pool Overflow	Cx Dropped Due To Reuse Pool Overflow	Total connections timed out in TCP MUX reuse pool.
Total Connections Closed by Server in TCP MUX Reuse Pool Overflow	Cx Dropped Due To Server Closed Cx In Reuse Pool	Total connections closed by server in TCP MUX reuse pool overflow.
Current Client Connections Waiting for Server Connection	Current Client Cx Waiting for Server Cx	Current client connections waiting for server connection
Total Responses Compressed	Total Responses Compressed	Total responses compressed.
Current Responses Being Compressed	Current Responses Being Compressed	Current responses being compressed.
Total Plain Text Bytes Before Compression	Input Bytes To Compress	Total plain text bytes before compression
Total Compressed Response Bytes	Total Responses Compressed	Total responses compressed.

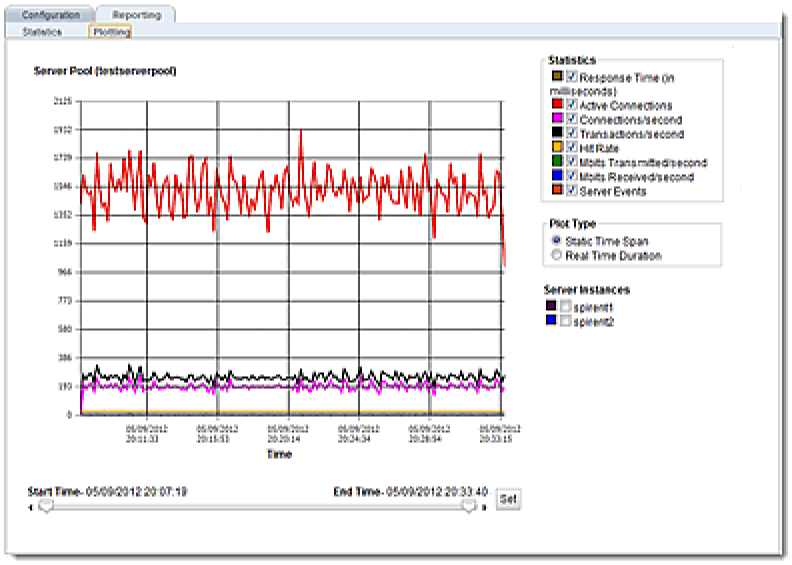
Server Instance Statistic Definitions

CLI Term	GUI Term	Definition
TOTALPRCSD	Total Connections	Connections Processed.
TOTALRESPPRCSD	Total Transactions	Responses Processed.
TIMESPENT	Total Time For Server Responses	Total time for server responses.
ACTIVECONX	Active Connections	Active Connections.
BYTERCVD	Bytes Received	Bytes received from peer.
BYTESEND	Bytes Sent	Bytes transmitted to peer.
TOTALSTKY	Total Sticky Records	Total sticky connections
CURRSTKY	Total Sticky Records	Current sticky record.
IDLECONXDROPED	Cx Dropped Due To Idle Timeout	Connections dropped for idle timeout.
STALECONXDROPED	Cx Dropped Due To Stale Timeout	Connections dropped for stale timeout.
FAILTHRICE	Same Server Selected After 3 Client Tries	Same server selected after 3 retries.
NEWFAILTHRICE	New Server Selected After 3 Client Tries	New server selected after 3 retries.
RSPPARSED	Number of Request Headers Parsed	Response headers parsed.
RSPFAILED	Number of Request Headers Failed Parsing	Responses failed header parsing.
RSPFAILHDR	Total Responses Dropped for Exceeding Header Limit	Responses dropped for exceeding header limit.
CLNTTO	Cx Dropped Due To Client Timeout	Connections dropped due to client timeout.
SRVRTO	Cx Dropped Due To Server Timeout	Connections dropped due to server timeout
CONNTO	Cx Dropped Due To Connect Timeout	Connections dropped due to connect timeout.
SELPERSIST	Server Selected By Cookie	No of Times Server Selected By Cookie.
SPLICE	Total Connections	Total client-server connections linked.
CURCLNTWAITQ	Current Client Cx Waiting for Server Cx	Client connections waiting for server connection.

CLI Term	GUI Term	Definition
REUSEOF	Cx Dropped Due To Reuse Pool Overflow	Connection dropped due to reuse pool overflow
REUSESRVR	Cx Dropped Due To Server Closed Cx In Reuse Pool	Connection dropped due to server closed connection in reuse pool.
REUSETO	Cx Dropped Due To Reuse Pool Timeout	Total connections timed out in TCP MUX reuse pool.
CURCOMP	Current Responses Being Compressed	Current responses being compressed.
TOTALCOMP	Total Responses Compressed	Total responses compressed.
INBYTECOMP	Input Bytes To Compress	Total plain text bytes before compression.
OUTBYTECOMP	Output Bytes After Compression	Total compressed response bytes.

The following is a graphical plot that can be displayed on the GUI. Select a server pool or server instance on the left navigational pane and click on the **Reporting** tab and then **Plotting**. The following will be displayed.

Sample Server Pool and Server Instance Graphical Plot

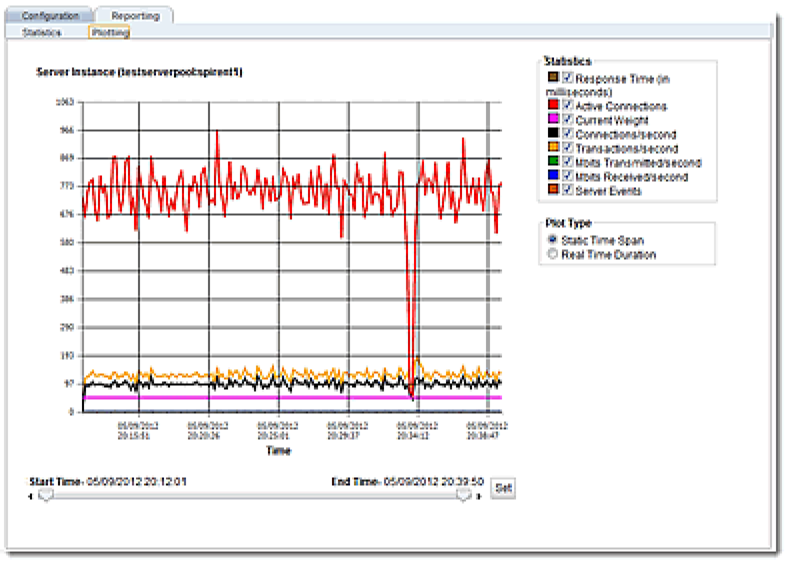


Statistics
- ☑ Response Time (in milliseconds)
- ☑ Active Connections
- ☑ Connections/second
- ☑ Transactions/second
- ☑ Hit Rate
- ☑ Mbits Transmitted/second
- ☑ Mbits Received/second
- ☑ Server Events

Plot Type
- ● Static Time Span
- ○ Real Time Duration

Server Instances
- ☐ spirent1
- ☐ spirent2

Start Time- 05/09/2012 20:07:19
End Time- 05/09/2012 20:33:40



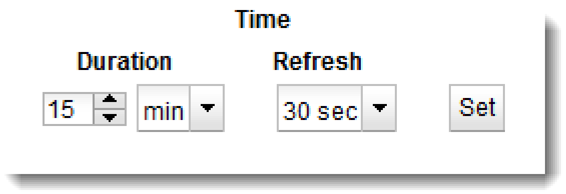
Statistics
- ☑ Response Time (in milliseconds)
- ☑ Active Connections
- ☑ Current Weight
- ☑ Connections/second
- ☑ Transactions/second
- ☑ Mbits Transmitted/second
- ☑ Mbits Received/second
- ☑ Server Events

Plot Type
- ● Static Time Span
- ○ Real Time Duration

Start Time- 05/09/2012 20:12:01
End Time- 05/09/2012 20:39:50

The specific types of statistics that are displayed are determined by the selections on the **Statistics** pane on the upper right corner of the GUI.Make selections based on the data that you require.

The **Plot Type** selection determines whether the display shown reflects a Static Time Span which is configured using the slider or whether a real time duration is display. If **Real Time Duration** is selected the slider controls will change to **Duration** and **Refresh** controls as shown below. In this case set the **Duration** of time in which you would like to review statistics and the **Refresh** rate desired.



Chapter 14

HTTP Caching

Sections in this chapter include:

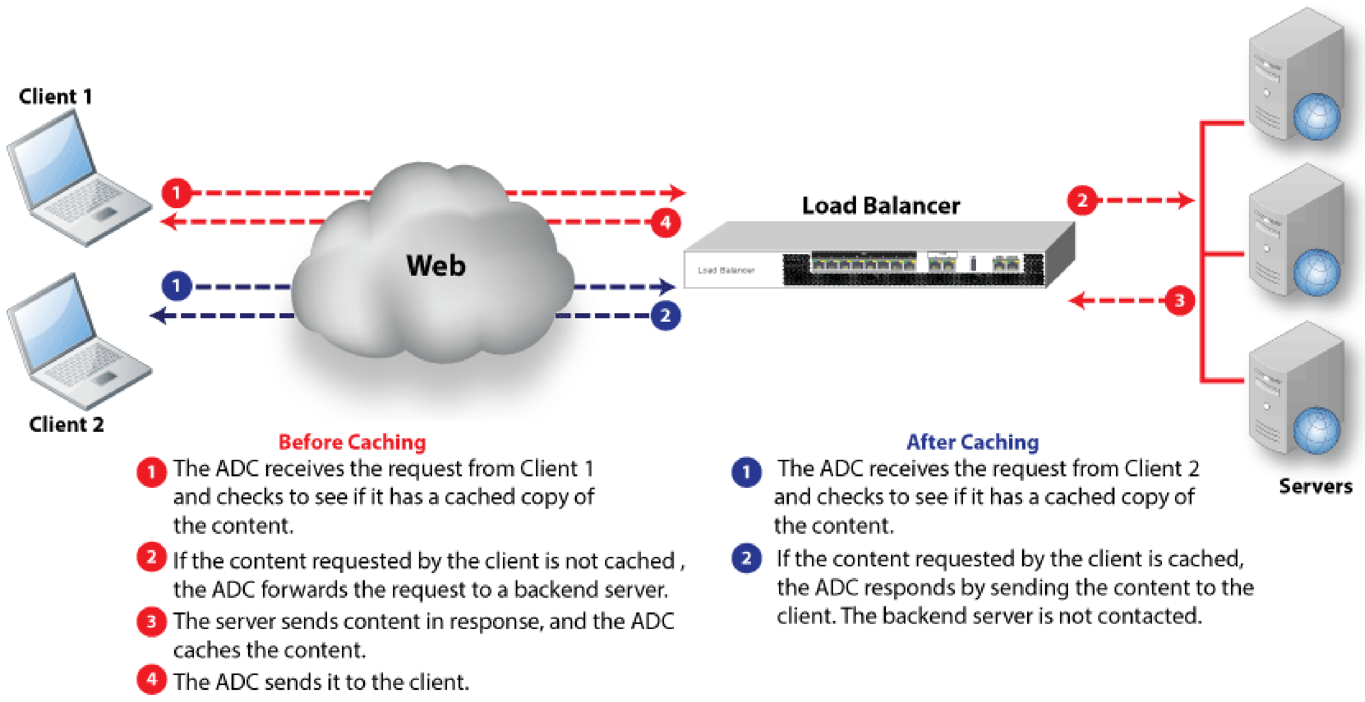
Caching Overview

Note - HTTP Caching is not available on legacy Equalizer GX models.

HTTP caching stores content previously received from servers in a special region of memory known as the cache. The goals of caching are:

- to provide a better client experience when accessing a cluster
- to reduce congestion and response times between the ADC and the servers behind it
- to reduce overall server load

The following illustration shows the impact of caching on the flow of content between clients and servers.


Client 1
Client 2
Web
Load Balancer
Servers
1
4
1
2
2
3
Before Caching
1 The ADC receives the request from Client 1 and checks to see if it has a cached copy of the content.
2 If the content requested by the client is not cached , the ADC forwards the request to a backend server.
3 The server sends content in response, and the ADC caches the content.
4 The ADC sends it to the client.
After Caching
1 The ADC receives the request from Client 2 and checks to see if it has a cached copy of the content.
2 If the content requested by the client is cached, the ADC responds by sending the content to the client. The backend server is not contacted.

HTTP Caching Functional Description

When a server response is received by the ADC, it is evaluated to determine if it is cacheable. Whether or not a specific response is cacheable is determined by certain attributes of the response:

- The status code of the response must be 200, 203, 300, 301, or 400.
- The file type of the response must be configured for caching. By default all file types are cacheable (text, graphics, videos, javascript, style sheets, etc.), but you can configure caching to exclude specific file types.
- The size of the item to be cached must not be above the configured maximum cached item size.

Once a response has been cached, it is classified as either *fresh* or *stale*. A response is fresh if its age hasn't exceeded the configured maximum cached item `age` parameter (**Maximum Cached Item Age** in the GUI); otherwise, the item is considered stale. When an item in the cache becomes stale, it means that it must be re-validated (and possibly re-fetched from the server) before it can be returned to the client.

Cached server responses are stored in memory. You can limit the amount of memory consumed by the cache using the CLI `max_size` parameter (**Maximum Cached Item Size** in the GUI). Once the total amount of cache memory reaches this limit, new entries will not be created until previous entries are deleted. Entries are chosen for deletion using the algorithm specified by the CLI `policy` parameter (**Replacement Policy** in the GUI).

Caching and Persistence

Persistence can have a substantial impact on caching performance. This is best explained by comparing what happens in the cache when persistence is not enabled vs. enabled.

Let’s say we have 3 servers in a cache-enabled server pool that is attached to a Layer 7 HTTP cluster, and all 3 servers return exactly the same content:

- When persistence is *disabled* on the cluster, then a single copy of any server response will be cached and will be returned to a client from the cache regardless of the server to which the client request was load balanced.
- When persistence is *enabled* on the cluster, then a copy of each server’s response will be cached even if the response is already cached due to another server returning that content. In this case, since there are 3 servers in the pool, you could have as many as 3 copies of the same content in the cache at any given time.

Because of the inefficiencies related to caching static content when persistence is enabled, it is recommended that you disable caching for static file types on server pools that are used by clusters that have persistence enabled. This can be done in either of two ways:

- Creating a match rule whose expression selects content that *should not* be cached, and attaching to the match rule a server pool that has caching *disabled*. This would be followed by a match rule that selects all other content and uses a server pool that has caching *enabled*.
- Creating a match rule whose expression selects content that *should* be cached, and attaching to the match rule a server pool that has caching *enabled*. This would be followed by a match rule that selects all other content and uses a server pool that has caching *disabled*.

See the section "Caching Specific File Types" on page 516 for more information.

Caching and Failover

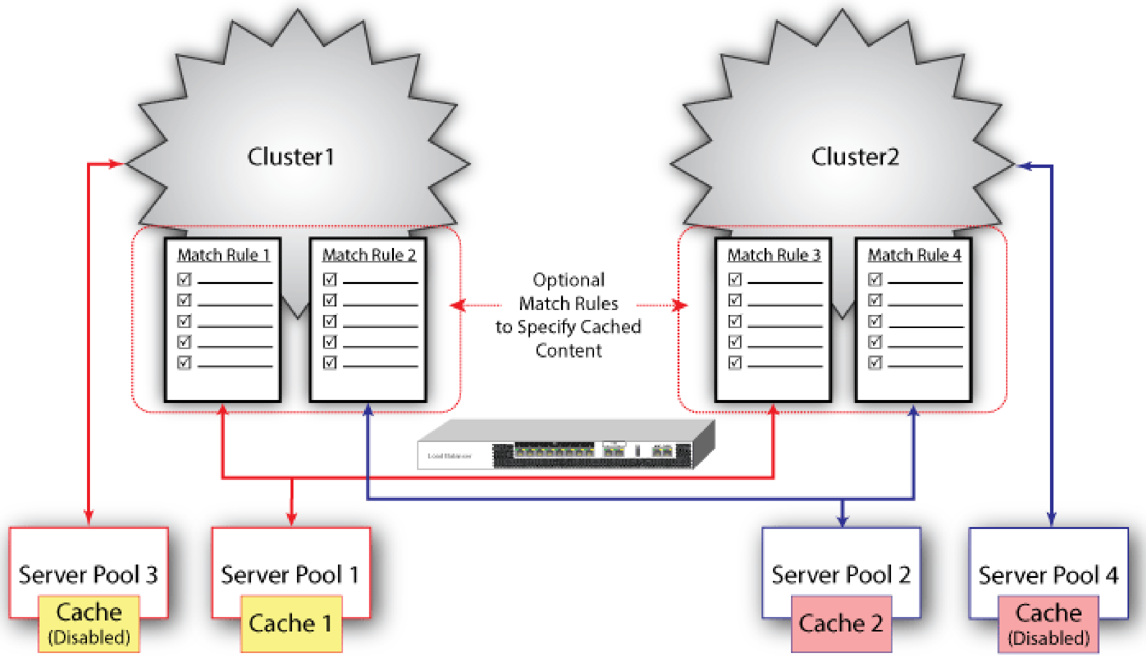
The caching configuration is synchronized across all failover peers. Cached content is not. When a failover occurs, the peer taking over as the primary peer for a failover group will begin caching server responses using the same settings that were used on the previous peer.

HTTP Caching Configuration Overview

As mentioned above, caching is configured on server pool objects. There is one HTTP cache allocated in memory for every server pool on the system for which caching is enabled. Server pools can be attached to both clusters and match rules:

- When you attach a cache-enabled server pool to a Layer 7 HTTP or HTTPS cluster, server responses delivered to clients through that cluster will be cached according to the caching parameter settings on that server pool.
- If a Layer 7 HTTP or HTTPS cluster has match rules, and a client request is load balanced to a server via a match rule, then it is the server pool attached to the match rule that caches the server responses (assuming caching is enabled on that server pool). Thus, match rules offer the ability to fine-tune your caching configuration by caching specific types of content in specific server pool caches.
- Cache-enabled server pools can be attached at the cluster level only, at the match rule level only, or both.

For example, the illustration below shows two clusters and their server pool/caching configuration:



In the example above:

- Both **Cluster1** and **Cluster2** have (different) server pools attached to them at the cluster level (**Server Pool 3** and **Server Pool 4**), and these server pools are configured with caching disabled.
- Both **Match Rule 1** (on **Cluster1**) and **Match Rule 3** (on **Cluster2**) send client requests to servers in **Server Pool 1**, and make use of its cache.
- Both **Match Rule 2** (on **Cluster1**) and **Match Rule 4** (on **Cluster2**) send client requests to servers in **Server Pool 2**, and make use of its cache.

When a server response is sent to either of these clusters from the back-end servers, the response is evaluated against the match rules as follows:

1. If the original client request matched the expression in Match Rule 1 or 3, then the response is stored in the cache for Server Pool 1.
2. If the original client request matched the expression in Match Rule 2 or 4, then the response is stored in the cache for Server Pool 2.
3. If the original client request did not match any of the match rule expressions above, then it is not cached, since caching is disabled on the server pool attached to the cluster.

Cache-enabled Server Pools and TCP/UDP Clusters

As stated above, caching is only performed for Layer 7 HTTP and HTTPS clusters. Caching is not supported for the following cluster types:

- Layer 4 TCP
- Layer 4 UDP
- Layer 7 TCP

If a cache-enabled server pool is attached to any of the above clusters types, the cache is simply ignored – that is, no server responses will be returned from the cache to the client and no server responses will be inserted into the cache.

The same cache-enabled server pool can be used in both a cluster that supports caching and one that does not; for example: in both a Layer 7 cluster and a Layer 4 TCP cluster. In this case, the cache will be used for the Layer 7 cluster and ignored for the Layer 4 cluster.

Using the CLI and GUI to Configure Caching

In the CLI, caching parameters are configured in the server pool context. In the GUI, caching parameters are set on the **Load Balance >** ***server pool name*** **> Configuration > Settings** tab. All caching-related parameters are set on server pools.

While match rules are used in specific situations to determine what content is cached, there are no caching-specific parameters set on match rule objects. It is recommended that you familiarize yourself with server pool configuration as well as match rule configuration before you configure caching. For more information on these topics refer to "Adding and Configuring Server Pools" on page 483 and "Creating a New Match Rule" on page 445

In order to configure caching, you must log in as a user that has the admin flag enabled. Refer to "User Flags" on page 1 for more information.

Caching Parameters

The table below describes the parameters used by the CLI and the GUI to configure caching on the server pool being modified:

Caching Parameters

GUI Parameter (CLI Parameter)	**Description**
Replacement Policy **(`policy`)**	This parameter sets the replacement policy for caching. When the cache is full, Equalizer chooses which items to discard to make room for the new ones using this policy. **Oldest**: The oldest entry (meaning it has been in the cache the longest) will be removed. **Largest**: The largest item will be removed. (**Default**: Oldest)
Maximum Cached Item Age **(`age`)**	This parameter specifies the maximum age of a cached item (in seconds) before it is counted as "stale". **Minimum**: 0 seconds **Maximum**: 86400 seconds (24 hours) (**Default**: 86400 seconds (24 hours)
Maximum Cached Item Size **(`max_object_size`)**	This parameter sets the maximum cached item size (in bytes). If a response is larger than this value, it will not be cached. **Minimum**: 0.bytes **Maximum**: 10485760 bytes (10MB)

Caching Parameters

GUI Parameter (CLI Parameter)	Description
Maximum Cache Size (**`max_size`**)	This parameter sets the maximum total size of the cache (in bytes), and is platform-dependent (see below). Once the total cache size reaches this size, new entries will not be created until some are deleted based on the replacement algorithm (**Replacement Policy**). **Minimum**: 1024 bytes **Maximum**: the maximum cache size (in bytes) is platform-dependent as show below. (**Default**:See Maximum below E370LX -- 104857600 bytes (100MB) E470LX -- 209715200 bytes (200MB) E670LX -- 307200 bytes (300MB) E970LX -- 524288000 bytes (500MB) Equalizer OnDemand-- 52428800 bytes (50MB) Note: HTTP Caching is not available on Equalizer GX systems.
Flags	
Disable (**`[!]disable`**)	Disables caching on the server pool. Since caching is disabled by default, you will need to *enable* it on the server pool using the CLI or the GUI. **`!disable`** enables caching on the server pool when using the CLI.

Enabling and Disabling Caching

Caching is *disabled* by default on all new server pools, and on all existing server pools after an upgrade. No content will be cached for a server pool until you explicitly enable it.

To enable caching using the CLI enter:

> `eqcli >` **`server pool`** ***`server_pool_name`*** **`cache flags !disable`**
>
> where:
>
> ***`server_pool_name`*** is the name of the server pool.

For example:

```
eqcli > srvpool srvpool1 cache flags !disable

eqcli > show srvpool srvpool1 cache
Cache is enabled.
Policy           : Oldest
Max Size         : 0
Max Object Size  : 2048
Age              : 86400
Flags            :
eqcli >
```

To disable caching using the CLI enter:

> `eqcli >` **`srvpool`** ***`server_pool_name`*** **`cache flags disable`**
>
> where:
>
> ***`server_pool_name`*** is the name of the server pool.

For example:

```
eqcli > srvpool srvpool1 cache flags disable

eqcli > show srvpool srvpool1 cache
Cache is disabled.
Policy           : Oldest
Max Size         : 0
Max Object Size  : 2048
Age              : 86400
Flags            : disabled
eqcli >
```

If caching is *disabled* on the server pool, a `Cache is disabled` message will be displayed and `disabled` will appear in the `Flags` field. If *enabled*, a `Cache is enabled` message will be displayed and the `Flags` field will be empty.

To enable caching using the GUI:

1. Log in to the GUI.
2. Select **Load Balance > Server Pool** on the left navigational pane and select the server pool on which you want to enable caching.
3. When the **Server Pool > Configuration > Required** tab is displayed on the right configuration pane, the **Disable** option is selected by default. Click on the checked box to remove the check and enable caching.
4. Click on **Commit** to save the option on the server pool.

To disable caching using the GUI:

Since caching is disabled by default, you do not need to disabled it unless it has been enabled.

1. Log in to the GUI
2. Select **Load Balance > Server Pool** on the left navigational pane and select the server pool on which you want to enable caching.
3. When the **Server Pool > Configuration > Required** tab is displayed on the right configuration pane, the **Disable** option is selected by default. Click on the checked box to remove the check and enable caching.
4. Click on **Commit** to save the option on the server pool.

All existing cached content will be flushed if you *disable* server pool caching.

Displaying Parameters of the Cache Attached to Server Pools

To view server pool caching parameteters in the CLI, enter:

```
eqcli > show srvpool server_pool_name cache
```

where:

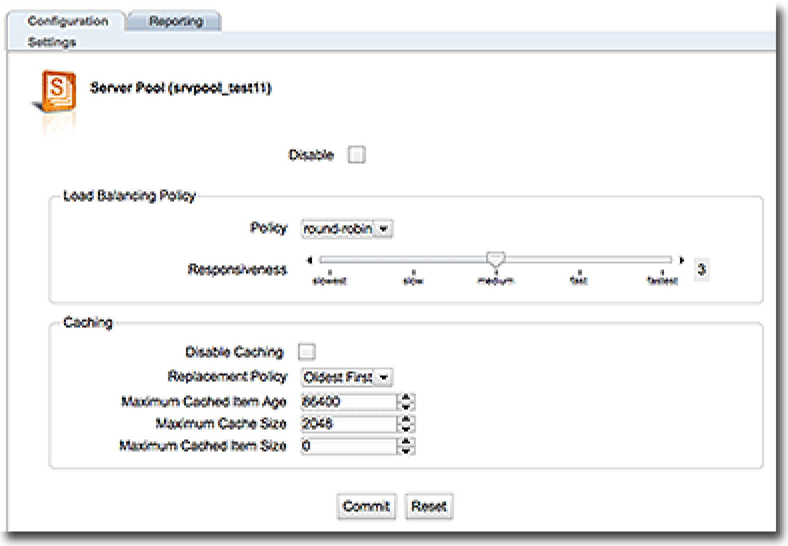
server_pool_name is the name of the server pool.

An example is shown below:

```
eqcli > show srvpool server_pool_name cache
Cache is enabled.
Policy              : Oldest
Max Size            : 0
Max Object Size     : 2048
Age                 : 86400
Flags               :
eqcli >
```

To view server pool caching parameters in the GUI:

1. Log in to the GUI.
2. Select **Load Balance > Server Pools** on the left navigational pane to expand the branch and select the server pool. The parameters will be displayed on the right configuration pane as shown below.



Server Pool (srvpool_test11)

Disable ☐

Load Balancing Policy

Policy: round-robin

Responsiveness: slowest, slow, medium, fast, fastest — 3

Caching

Disable Caching ☐

Replacement Policy: Oldest First

Maximum Cached Item Age: 86400

Maximum Cache Size: 2048

Maximum Cached Item Size: 0

Note - If caching is *disabled*, the **Replacement Policy, Maximum Cached Item Age**, **Maximum Cache Size**, and **Maximum Cached Item Size** parameters will not be displayed.

Configuring Caching Parameters

All of the caching configuration parameters have default values as shown in the section "Caching Parameters" on page 509. Should you need to configure non-default values, you can do so using either the CLI or the GUI as shown in the following sections.

Using the CLI

Configure caching parameters in the CLI using the following format.

```
eqcli >server pool server_pool_name cache parameter parameter_value
```

where:

parameter can be `policy`, `age`, `max_object_size`, or `max_size`.

parameter_value is the value assigned to the parameter.

For example:

```
eqcli >server pool srvpool1 cache policy oldest
```

You could also configure caching parameters in the server pool context or cache context. For example:

```
eqcli sp-srv*> cache policy oldest
```

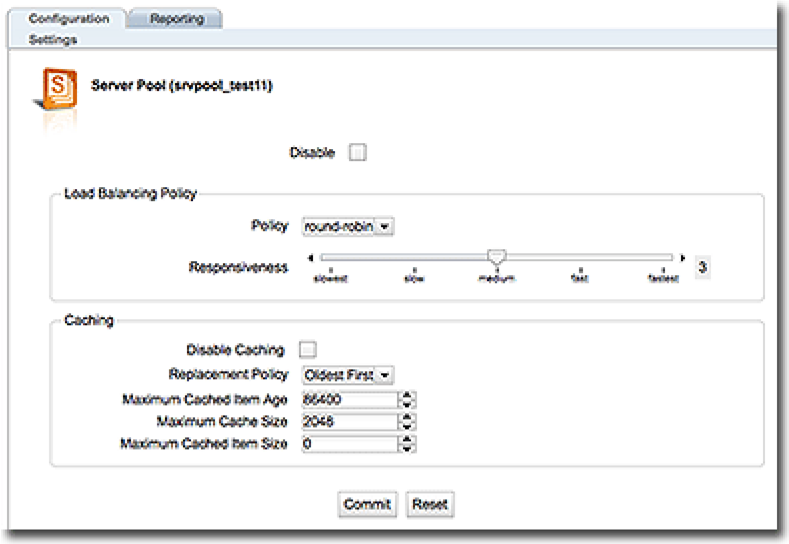
or

```
eqcli sp-srv*-cache> policy oldest
```

Using the GUI

Caching parameters are configured on the server pool configuration screen on the GUI.

1. Log in to the GUI.
2. Access the **Server Pool > Configuration > Settings** display for the caching server pool.



Server Pool (srvpool_test11)

Disable ☐

Load Balancing Policy

Policy: round-robin

Responsiveness: slowest – slow – medium – fast – fastest: 3

Caching

Disable Caching	☐
Replacement Policy	Oldest First
Maximum Cached Item Age	86400
Maximum Cache Size	2048
Maximum Cached Item Size	0

3. Click on the **Reset** button to remove anything that you've entered and return the screen to the default display.
4. Click on **Commit** to save the parameters.

Caching Specific File Types

Match rules provide finer granularity in caching content in that they provide a means of excluding or including certain file types in the cache. For example, you may want to cache only graphical files (e.g.,.png, .gif, ,jpg, .jpeg, .ico), video files (e.g.,.mp4, .flv, .mov, .mpeg, .swf), javascript files, or cascading style sheets (css). Using match rules, you can customize which server content is cached, based on your needs.

To demonstrate this feature, we'll configure an example that will tell Equalizer to cache only graphical content with *.png, *.jpg, or *.gif file extensions. We'll configure a cluster with a match rule that uses expressions that identify file suffixes. When a filename_suffix (string) is used in the expression, it will evaluate to true if the string argument that you specify is a suffix of the file name portion of the URI path- in this case *.png, *.jpg, or *.gif . The content specified by the match rule expression will then be cached.

Using the CLI

1. Verify that caching is *disabled* on a server pool so that caching will not begin until you configure a match rule. Refer to"Enabling and Disabling Caching" on page 511 for a description of the procedure.
2. Create a match rule (using an existing cluster of your choice) and add an appropriate server pool to it at the same time. Use the following format:

   ```
   eqcli > cluster clustername match match_name.srvpool server_pool_name
   ```

   For example:

```
eqcli > cluster cluster_http match test srvpool srvpool1
```

3. Display the match rule to confirm that it is disabled. We do not want this match rule to be used until the configuration is complete.

```
eqcli > show cluster_http match test
Match Rule Name          : test
Next Match Rule          : none
Cluster Name             : cluster_http
Server Pool              : srvpool1
Responder                :
Cookie Path              :
Cookie Domain            :
Cookie Age               : 0
Cookie Generation        : 0
Persist Type             : coyote_cookie_2
Minimum Compression      : 1024
Compression mime_type    :
text/*:application/msword:application/postscript:
application/rtf:application/x-csh:
application/x-javascript:application/x-sh:application/
x-shar:application/xtar:
application/x-tcl:
application/xslt+xml:audio/midi:audio/32kadpcm:audio/xwav:
image/bmp:image/tiff:image/x-rgb
Flags : disable
Expression :
Config Header Edit Script :

eqcli cl-clu*-ma-test>
```

4. Configure a match rule expression. The expressions that you will need to configure are **`filename_suffix (\".gidf\")`**, **`filename_suffix(\".jpg\")`**, and **`filename_suffix (\".png\")`**. As you can see in the example below, "`||`" are logical "OR" operators added to the expression. This means that the match rule will parse .gif OR .jpg OR .png files.

```
eqcli > cluster cluster_http match test expression filename_suffix
(\".gif\")||filename_suffix(\".jpg\")||filename_suffix(\".png\")

eqcli > commit
```

5. Verify that the new match rule includes the expression that you created. The expression used in our example is highlighted below.

```
eqcli > show cluster_http match test
Match Rule Name          : test
Next Match Rule          : none
Cluster Name             : cluster_http
Server Pool              : srvpool1
Responder                :
Cookie Path              :
Cookie Domain            :
Cookie Age               : 0
Cookie Generation        : 0
Persist Type             : coyote_cookie_2
Minimum Compression      : 1024
Compression mime_type    :
text/*:application/msword:application/postscript:
application/rtf:application/x-csh:
application/x-javascript:application/x-sh:application/
x-shar:application/xtar:
application/x-tcl:
application/xslt+xml:audio/midi:audio/32kadpcm:audio/xwav:
image/bmp:image/tiff:image/x-rgb
Flags : disable
Expression :
filename_suffix(\".gif\") || filename_suffix(\".jpg\") || filename_suffix
(\".png\")
Config Header Edit Script :

eqcli cl-clu*-ma-test>
```

6. Enable caching on the server pool. For example:

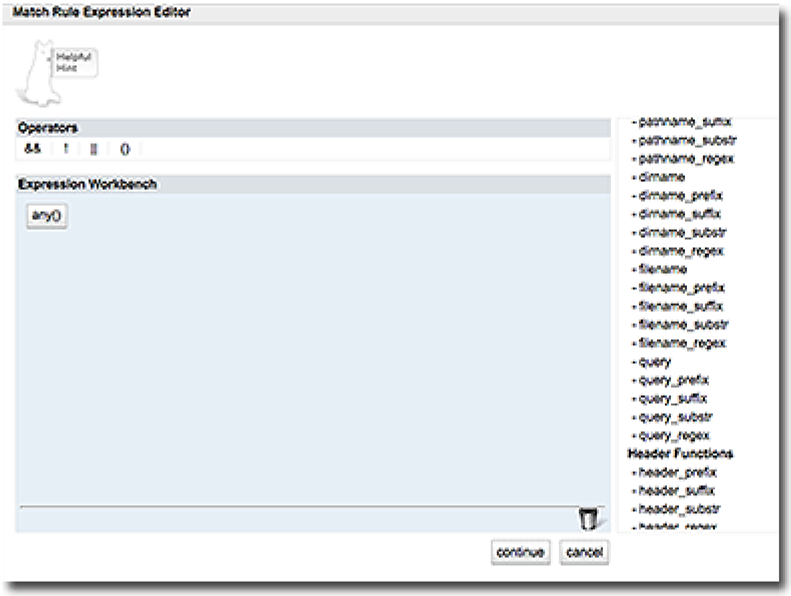
```
eqcli > srvpool srvool1 cache !disable

eqcli > show srvpool srvpool1 cache
Cache is enabled.
Policy           : Oldest
Max Size         : 3000
Max Object Size  : 2048
Age              : 86400
Flags            :
eqcli >
```

7. Enable the match rule.

```
eqcli cl-clu*-ma-test> flags enable
eqcli cl-clu*-ma-test> flags commit
```

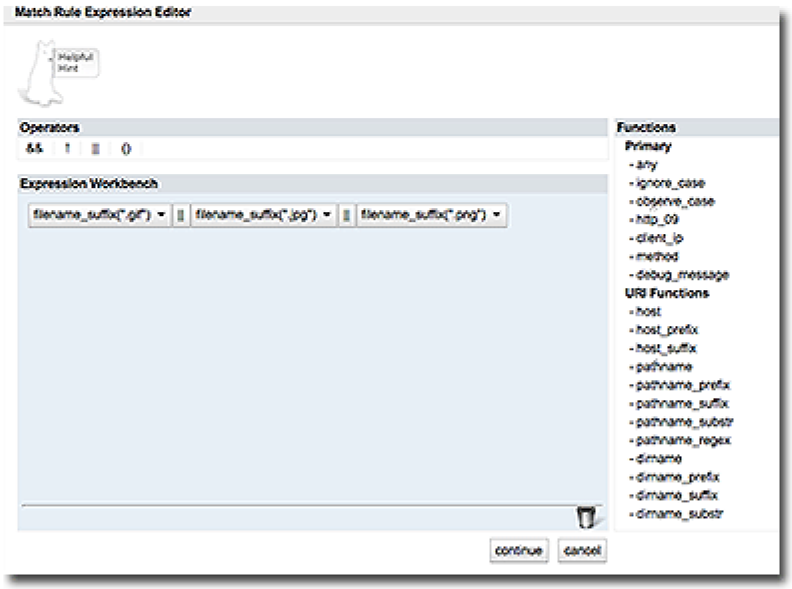
Using the GUI

1. Log in to the GUI.
2. Verify that caching is disabled on the caching server pool so that caching will not begin until you configure a match rule. Refer to "Enabling and Disabling Caching" on page 511 for a description of the procedure.
3. Select the HTTP cluster to which the caching server pool is attached. Right click on it and select **Add Match Rule**.
4. Enter a **Match Rule Name** and click on **Commit**. The new match rule will be displayed on the cluster branch on the left navigational pane and the match rule **Configuration > Required** screen will appear on the right.
5. Verify that the **Disable** option is enabled for the match rule. We don't want this match rule to be used until the configuration is complete.
6. Select the caching server pool from the **Server Pool** drop down list. If you do not select the caching server pool, the match rule will not be used for caching.
7. Click on the **Expression Editor** button to display the **Match Rule Expression Editor**. Drag the **any()** button to the trash can icon below.



8. Select **filename_suffix** from the URI functions on the right and drag it to the **Expression Workbench**. Click on the **filename_suffix** button on the **Workbench** and add ".gif" in the space provided.
9. Drag the || button from the **Operators** pane above the **Expression Workbench** to the space after your first **filename_suffix(".gif")** expression. This is a logical "OR" operator.This is added to the expression so that the match rule will parse .gif OR .jpg OR .png files.

10. Repeat steps 8 and 9, for both .jpg and .png file suffixes. When complete, your expression should be as follows:



filename_suffix(".gif") || filename_suffix(".jpg") || filename_suffix(".png")

11. Click on **Continue** to return to the match rule **Configuration > Required** screen.
12. Click on **Commit** to save the match rule.
13. Verify that caching is enabled on the caching server pool used by the match rule. Select **Load Balance > Server Pools >** ***server pool name*** **> Configuration** tab. Verify that **Disable Caching** is NOT checked in the caching group.
14. Enable the Match Rule. Select **Load Balance > Clusters >** ***cluster name*** **>** ***match rule name*** on the left navigational pane to display the match rule **Configuration** tab. Uncheck the **Disable** option.

Testing the Caching Configuration

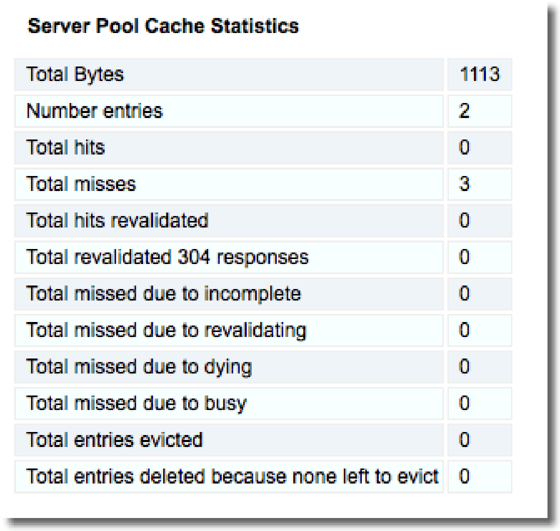
Test the match rule and caching configuration above by sending traffic through a cluster. Using the statistics displays in the GUI and CLI, (See "Show Caching Statistics" on page 523) verify that, for example, retrieving a graphical file increases the relevant counters and a .txt file does not.

1. Verify that there are ".txt", ".gif", ".gif", and ".jpg" files on a server. For this example, we will retrieve "Test.jpg" and "Test.txt" from a server.
2. Add the server to the configuration on the **Server** branch on the left navigational pane.on the GUI. (Refer to "Adding and Modifying Servers" on page 529.)
3. Add the server instance to the caching server pool.and enable caching. (Refer to "Enabling and Disabling Caching" on page 511
4. Select the cluster from the left navigational pane and verify that the caching server pool is selected from the **Server Pool** drop down list.
5. View and make note of your current caching statistics. Select the caching server pool from the left navigational pane and click on the **Reporting** tab on the right. Make a note of the current **Number Entries**, **Total** hits, and **Total misses** statistics.
6. Open a new browser window, enter **http://*cluster_ip_address*/Test.txt** in your browser's address bar and **ENTER**.
7. Select **Load Balance > Server Pools > *server pool name*** on the GUI left navigational pane.
8. Select the **Reporting** tab on the right configuration pane and note the **Server Pool Caching Statistics**. When you entered the "Test.txt", Equalizer made the request to the server. Since the match rule in the configuration specifies that you want to cache ".jpg", ".png", or ".gif" files only, the .txt file was not cached and the **Number entries** or **Total hits** caching statistics did not increase from the values that you noted from step 5. Also, the **Total misses** statistic should increase as the "Test.txt" file did not meet the match rule stipulations for caching.
9. Open a new browser window, enter **http://*cluster_ip_address*/Test.jpg** in your browser's address bar and **ENTER**.
10. Select **Load Balance > Server Pools > *server pool name*** on the GUI left navigational pane.

11. Select the **Reporting** tab on the right configuration pane and note the **Server Pool Caching Statistics**. When you entered "Test.jpg" on your browser's address bar, Equalizer made the request to the server. Since the match rule in the configuration specifies that you want to cache only .jpg, .png, or .gif graphical files, "Test.jpg" was cached and the **Number entries** or **Total hits** caching statistics increased.

    - If an item *was not* previously cached, the **Number entries** statistic will increase.
    - If an item *was* previously cached, the **Number entries** statistic will remain the same, however, the **Total hits** statistic will increase as the request matched a cached entry in the ADC.

    Each time that you attempt to retrieve "Test.jpg" the **Number entries** caching statistic will remain the same and the **Total hits** statistic will increase. An example is shown below. Refer to "Show Caching Statistics" on the facing page for descriptions of the statistics.



Server Pool Cache Statistics

Total Bytes	1113
Number entries	2
Total hits	0
Total misses	3
Total hits revalidated	0
Total revalidated 304 responses	0
Total missed due to incomplete	0
Total missed due to revalidating	0
Total missed due to dying	0
Total missed due to busy	0
Total entries evicted	0
Total entries deleted because none left to evict	0

Show Caching Statistics

The caching statistics displayed for a server pool represent all requests that have been received/responded since the cache was instantiated, either when it was configured or the system started up or rebooted. These statistics are *not* saved between system reboots.

Statistic	**Description**
Total bytes	The total number of bytes of response data stored in the cache. Does not include request header size.
Number entries	The total number of entries currently cached. It is possible that some entries may not be usable if they are: incomplete - the ADC started creating an entry but never finished, possibly because the server didn't respond invalid - data exists for the entry, however the headers disallow the entry from being returned without revalidation deleting - an entry may be in the process of being deleted, but hasn't been as yet
Total hits	A cache *hit* is recorded when cached data can be returned to the client.
Total misses	The total number of unsuccessful cache requests since start of cache service.
Total hits revalidated	A cached entry was found, however, was not fresh - we had to check with the server to revalidate.
Total revalidated 304 responses	The total number of times that a server responded with a 304 response. A returned 304 status code means the content is still fresh.
Total missed due to incomplete	The total number of times that a cached entry was found, however, it was not complete so it could not be used.
Total missed due to revalidating	The total number of times that a cached entry was found, however, it was in the process of being revalidated for another client and could not be used.
Total missed due to dying	The total number of times that a cached entry was found, however,it was in the process of being removed and could not be used.
Total missed due to busy	The total number of times that a cached entry was found, however, could not be used because the system was too busy to return it.
Total entries evicted	The number of cached items that have been evicted from the cache in order to make room for newer cached items.
Total entries deleted because none left to evict	When an entry is evicted because the cache is out of space, it is marked for deletion - but doesn't actually get deleted until the system has time to do so. If every entry has been marked for deletion, but the cache is still full, there will be no room to create more entries.

To display caching statistics for an enabled server pool using the CLI enter:

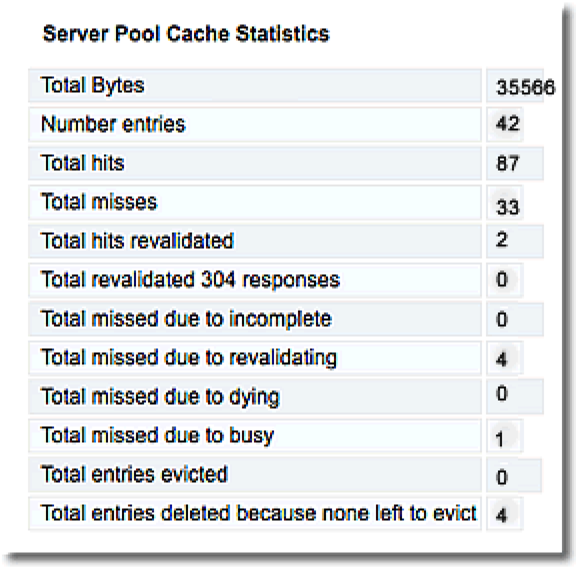
> eqcli > **server pool *serverpoolname* cache stats**
>
> where:
>
> ***serverpoolname*** is the name of the server pool.

The following is an example of the display.

```
eqcli > srvpool srvpool1 cache stats
Statistics                                          Value
Total bytes                                       : 35566
Number entries                                    : 42
Total hits                                        : 87
Total misses                                      : 33
Total hits revalidated                            : 2
Total revalidated 304 responses                   : 2
Total missed due to incomplete                    : 2
Total missed due to revalidating                  : 4
Total missed due to dying                         : 0
Total missed due to busy                          : 1
Total entries evicted                             : 7
Total entries deleted because none left to evict  : 4
eqcli >
```

To display the caching statistics for an enabled server pool using the GUI:

1. Log in to the GUI.
2. Select **Load Balance > Server Pool** from the left navigational pane.
3. Select the server pool and then select **Reporting > Statistics** on the right configuration pane. The following will be displayed on the right side of the display.



Server Pool Cache Statistics

Total Bytes	35566
Number entries	42
Total hits	87
Total misses	33
Total hits revalidated	2
Total revalidated 304 responses	0
Total missed due to incomplete	0
Total missed due to revalidating	4
Total missed due to dying	0
Total missed due to busy	1
Total entries evicted	0
Total entries deleted because none left to evict	4

Chapter 15

Managing Servers

Sections within this chapter include:

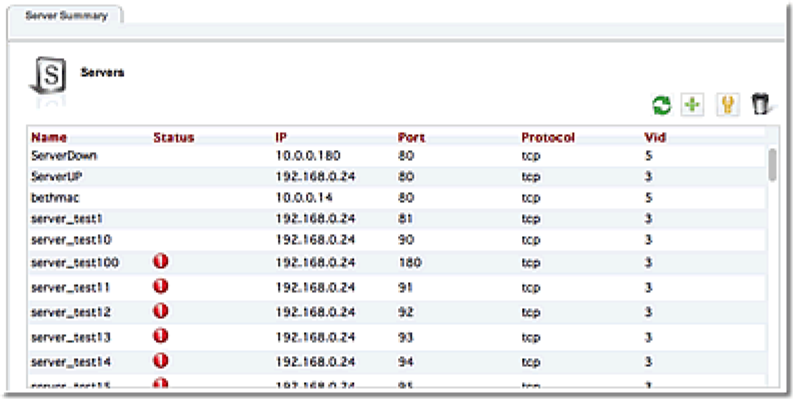
Server Summary

A summary of the servers configured on your Equalizer be viewed using either the GUI or the CLI.

Server Summary Using the GUI

The **Server Summary** screen displays the names of configured servers, **IP** address, **Ports**, **Protocol**, **VID** ,and associated **Server Pools.**

1. Log in to the GUI.
2. Select the **Load Balance** > **Servers** to display the **Server Summary** screen.

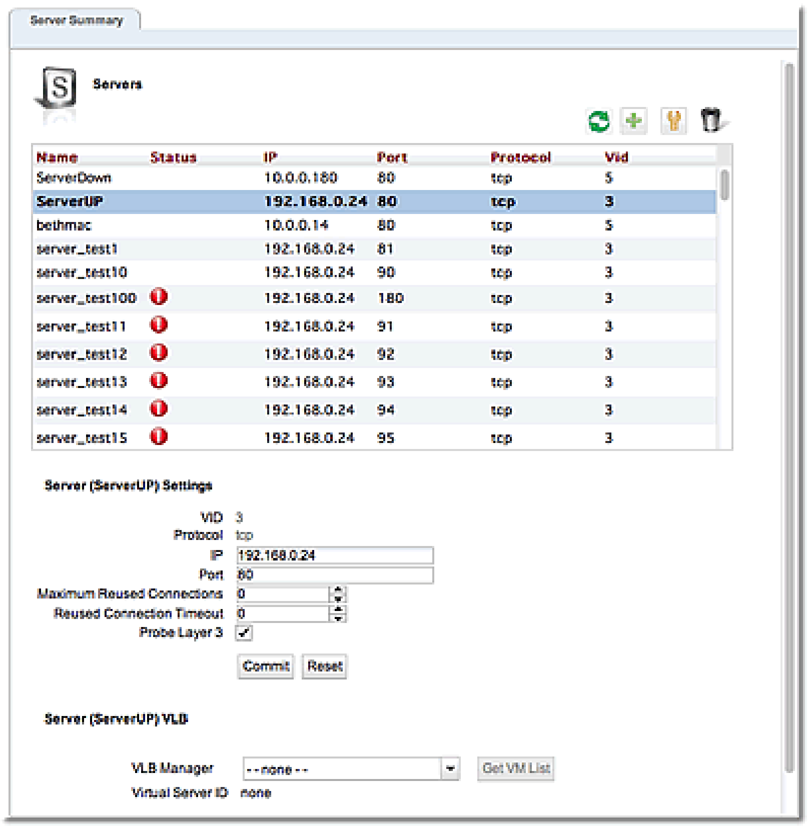


Name	Status	IP	Port	Protocol	Vid
ServerDown		10.0.0.180	80	tcp	5
ServerUP		192.168.0.24	80	tcp	3
bethmac		10.0.0.14	80	tcp	5
server_test1		192.168.0.24	81	tcp	3
server_test10		192.168.0.24	90	tcp	3
server_test100	●	192.168.0.24	180	tcp	3
server_test11	●	192.168.0.24	91	tcp	3
server_test12	●	192.168.0.24	92	tcp	3
server_test13	●	192.168.0.24	93	tcp	3
server_test14	●	192.168.0.24	94	tcp	3

From this screen you can add a new server by clicking on "+" .

You can delete a server by selecting a server and clicking on .

You can modify server configuration by selecting a server and clicking on . This will expand the window activate the server **Settings** area as shown below. Refer to "Adding and Modifying Servers" on page 529



Name	Status	IP	Port	Protocol	Vid
ServerDown		10.0.0.180	80	tcp	5
ServerUP		192.168.0.24	80	tcp	3
bethmac		10.0.0.14	80	tcp	5
server_test1		192.168.0.24	81	tcp	3
server_test10		192.168.0.24	90	tcp	3
server_test100	●	192.168.0.24	180	tcp	3
server_test11	●	192.168.0.24	91	tcp	3
server_test12	●	192.168.0.24	92	tcp	3
server_test13	●	192.168.0.24	93	tcp	3
server_test14	●	192.168.0.24	94	tcp	3
server_test15	●	192.168.0.24	95	tcp	3

Server (ServerUP) Settings

VID	3
Protocol	tcp
IP	192.168.0.24
Port	80
Maximum Reused Connections	0
Reused Connection Timeout	0
Probe Layer 3	✓

Server (ServerUP) VLB

VLB Manager	-- none --
Virtual Server ID	none

Server Summary Using the CLI

The server summary shown below displays a summary listing of all of the servers, protocol, IP addresses, ports, and any options (flags) configured on your Equalizer.

Enter the following:

```
eqcli > show server
Name        Protocol  IP Address  Port  Flags
srv1        tcp       2.4.6.8     80
srvudp      udp       3.6.9.12    80
servertcp   tcp       2.5.7.9     80
testing123  tcp       6.7.8.9     80
12000003: You have 1 pending alert notification.
eqcli >
```

To display the summary for a specific server use the format: `eqcli >` **`show server`** ***`servername`***, where ***`servername`*** is the name of the server that you want to view. For example:

```
eqcli > show server srv1
This server is enabled.
Server Name                 : srv1
IP Address                  : 2.4.6.8
Port                        : 80
Protocol                    : tcp
VID                         : 2
Max Reuse Connections       : 0
Reuse Connections Timeout   : 0
VLB Manager                 :
UUID                        :
eqcli >
```

Adding and Modifying Servers

Servers can be added and modified using either the GUI or the CLI.

Parameters

The table below shows the parameters, values, and flags used in the configuration Servers.

GUI Parameter (CLI Parameter)	Description
IP **(`ip`)**	The dotted decimal IP address of the server. This is the address Equalizer uses to communicate with the server.
Port **(`port`)**	Enter the numeric port number on the Equalizer to be used for traffic between Equalizer and the server. The default is port 80. (Note that in *Layer 7 HTTPS* clusters, the server port should be set to something other than 443 since Equalizer communicates with servers in an HTTPS cluster via HTTP.) For L4 UDP and L4 TCP protocol clusters, a cluster *port range* can be defined. These are the ports on the Equalizer to be used to send traffic to the server pool in the cluster. Port ranges allow Equalizer users to create a single cluster to control the traffic for multiple, contiguous ports. The **Port** defined for a *server* in the cluster for which a port range is defined indicates the port on the server that starts the range of ports to be opened.
Maximum Reused Connections **(`max_reuse_conn`)**	By default, the **Maximum Reused Connections** server option is set to 0, which means that Equalizer will route traffic to the server whenever the server is selected by the current load balancing settings. If **Maximum Reused Connections** is set to a value greater than 0, then Equalizer limits the total number of simultaneously open connections to the server to that value. This restriction applies regardless of the persistence options set on the cluster. When a server reaches the specified limit, requests will not be routed to that server until the number of active connections falls below the limit.Typical reasons to set a maximum number of connections include: •Implementing a connection limit that is required due to software limitations, such as an application that can service a limited number of concurrent requests. •Implementing license restrictions that are not enforced by software; such as limiting the number of active connections to an application that is licensed for a limited number of concurrent connections. •Setting a threshold that will limit resource utilization on the server.
Reused Connection Timeout **(`reuse_conn_to`)**	The number of seconds after which a connection record for an idle server connection in the reusable connection pool is removed, and the connection closed. The default value is **0** seconds, which means that records in the reusable connections pool never expire.

Adding and Modifying Servers Using the GUI

Adding Servers

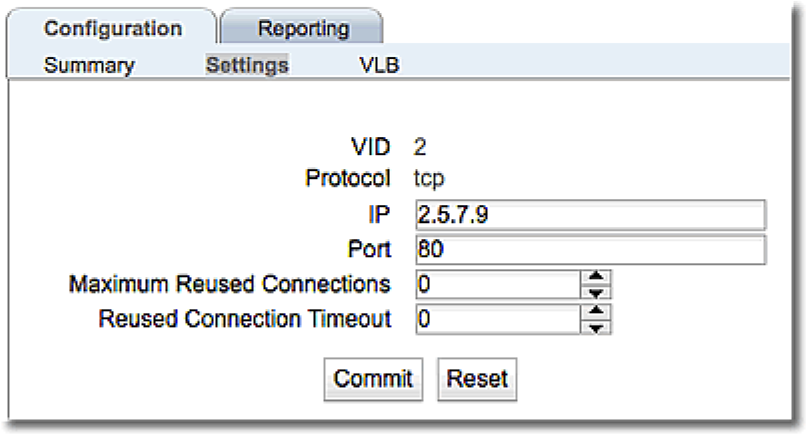
Perform this procedure once for each real server that you want to add to Equalizer.

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Load Balance** configuration tab if it is not already selected.
3. Right-click on **Servers** and select **Add Server**. This will open the **Add Server** dialogue as follows:
4. Select **UDP** or **TCP** from the drop down list and enter a descriptive **Server Name**.
5. Enter an **IP** address for the server.
6. Enter a **Port** for the server. For HTTP, this is typically port 80. For HTTPS, port 443. For generic TCP and UDP services, use the port appropriate for that service. Whatever port you enter, it must match the port set on the real server.
7. Click on **Commit** to confirm the values you selected. You will be taken to the Server Configuration tab and the new server will appear on the Server branch on the left navigational pane. The server IP address and **Port** will be visible in the **Server Configuration** screen.

Modifying Servers

The configuration tabs for a server are displayed automatically when a server is added to the system, or by selecting the server name from the left navigational pane.

1. Log into the GUI using a log in that has at least **write** access for the cluster that contains the server.
2. In the left navigational pane, select the **Load Balance > Servers >** ***Server Name*** **> Configuration > Settings** to display the following.



VID 2
Protocol tcp
IP 2.5.7.9
Port 80
Maximum Reused Connections 0
Reused Connection Timeout 0

Note - *For servers in Layer 7 HTTPS clusters*, set **Port** to something other than 443, since Equalizer communicates with the servers via HTTP. In many configurations, it is set to the server **Port**.

Clicking on the **Reset** button will remove anything that you've entered and return the screen to the default display. If you made any changes to the default configuration values, click on the **Commit** button to save your changes.

Adding and Modifying Servers using the CLI

Adding Servers

Perform this procedure once for each real server that you want to add to Equalizer.

Enter the following:

```
eqcli > server [server name] proto tcp ip xxx.xxx.x.x port xx
```

where:

`proto` - is the server protocol

`ip` - is the dotted decimal IP address of the server. This is the address Equalizer uses to communicate with the server.

`port` - is the numeric port number on the Equalizer to be used for traffic between Equalizer and the server. The default is port 80. (Note that in *Layer 7 HTTPS* clusters, the server port should be set to something other than 443 since Equalizer communicates with servers in an HTTPS cluster via HTTP.) For L4 UDP and L4 TCP protocol clusters, a cluster *port range* can be defined. These are the ports on the Equalizer to be used to send traffic to the server pool in the cluster. Port ranges allow Equalizer users to create a single cluster to control the traffic for multiple, contiguous ports. The **Port** defined for a *server* in the cluster for which a port range is defined indicates the port on the server that starts the range of ports to be opened.

Note - *For servers in Layer 7 HTTPS clusters*, set **Port** to something other than 443, since Equalizer communicates with the servers via HTTP. In many configurations, it is set to the server **Port**.
The **server agent port**, set on the cluster, remains a separate port that is used only for server agent communication.)

Modifying Servers

Servers can be configured in the CLI either globally or in cluster context. Enter parameters using the following format.

```
eqcli > server servername parameter value
```

where:

servername - is the name of the server

parameter - is the parameter to be configured

value - is the value associated with the parameter.

For example:

```
eqcli > server srv1 max_reuse_conn 5
```

Refer to "Parameters" on page 529 above for details about server parameters.

Refer to "Server Commands" on page 208 for additional information on using server commands in the CLI.

Server Software Configuration

Please observe the following guidelines and restrictions when configuring the software on your servers:

Spoof

If the **spoof** flag is turned on for a cluster (the default), you should configure your network topology so that Equalizer is the gateway for ***all*** traffic for its virtual clusters. In most cases, this means that each server in a cluster should be configured to use Equalizer as its default gateway, so that all packets that come through Equalizer from clients will pass back through Equalizer and then to the clients.

You do *not* need to configure Equalizer as the gateway for the servers if you have *disabled* the IP **spoof** flag for the cluster.

Header Limit

Server responses (and client requests) must contain 64 or fewer headers; any packet that contains more than 64 headers is dropped by Equalizer (along with the connection), and a message like the following is printed to Equalizer's event log:

```
Warning: Dropping connection from ip-address -- too many headers
```

Make sure that your server software is configured to return 64 headers or less in any response it sends back through Equalizer. Be aware, however, that this has no effect on the client side; any packets from the client with more than 64 headers will still be dropped by Equalizer (and a warning appended to the event log). In most cases, client requests do not include that many headers.

Adjusting a Server’s Initial Weight

Equalizer uses a server’s initial weight as the starting point for determining the percentage of requests to route to that server. As Equalizer gathers information about the actual performance of a server against client requests, it adjusts the server’s current weight so that servers that are performing well receive a higher percentage of the cluster load than servers that are performing at a slower rate.

When you install servers, set each server's initial weight value in proportion to its “horsepower.” All the initial weights in a cluster do not need to add up to any particular number; *it’s the ratio of the assigned server weight for a server to the total of all the server weights that determines the amount of traffic sent to a server*.

For example, you might assign a server with 4 dual-core 64-bit processors operating at 3.40GHz a value of 100 and a server with 2 dual-core 64-bit processors operating at 1.86GHz a value of 50. The first server will initially receive approximately 66% (100 divided by 150) of the traffic. The second server will initially get about 33% (50 divided by 150) of the traffic. It’s important to note that setting the initial weights of these servers to 100 and 50 is equivalent to setting the initial weights to 180 and 90.

Values for server weights can be in the range 0-200, with 0 meaning that no new requests will be routed to the server, essentially disabling the server for subsequent requests. In general, you should use higher initial weights. When the load balancing policy is ***not*** set to round robin or static weight, using higher initial weights will produce finer-grained load balancing. Higher weights enable Equalizer to adjust server weights more gradually; increasing the weight by 1 produces a smaller change if the starting weight is 100 than it does if the starting weight is 50.

However you set the initial weights, Equalizer will adjust the weight of servers dynamically as traffic goes through the cluster. Dynamic server weights might vary from 50-150% of the statically assigned values. To optimize cluster performance, you might need to adjust the initial weights of the server instance in a server pool based on their performance.

Note - Equalizer stops dynamically adjusting server weights if the load on the cluster drops below a certain threshold. For example, if web traffic slows significantly at 4:00 AM PST, Equalizer will not modify server weights until traffic increases again. Because a server’s performance characteristics can be very different under low and high loads, Equalizer optimizes only for the high-load case. Keep this in mind when you configure new Equalizer installations; to test Equalizer’s ALB performance, you’ll need to simulate expected loads.

Setting Initial Weights for Homogenous Clusters

If all the servers in a cluster have the same hardware and software configurations, you should set their initial weights to the same value initially. We recommend that you use a initial weight of 100 and set the load-balancing response parameter to *medium*.

As with any new configuration, you will need to monitor the performance of the servers under load for two to three hours. If you observe that the servers differ in the load they can handle, adjust their initial weights accordingly and again monitor their performance. You should adjust server weights by small increments; for example, you might set the initial weight of one server to 110 and the other to 90. Fine-tuning server weights to match each server’s actual capability can easily improve your cluster’s response time by 5 to 10%.

Note - A change to a server’s initial weight is reflected in cluster performance only after Equalizer has load balanced a significant number of new client requests for up to 30 minutes against the cluster in which the servers reside. When testing initial weights, it is most useful to use a load-generating tool to run typical client requests against the cluster to determine appropriate server initial weights.

Maximum Connections Limits, Responders, and Hot Spares

When a maximum connections limit is set on all the servers in a cluster, it is often desirable to define either a responder or a hot spare server for the cluster, so that any attempted connections to the cluster that occur after the **Maximum Reused Connections** limit has been reached are directed to the responder or hot spare instead of being refused or sent to the server anyway because of a persistent connection.

In general, a Responder is easier to configure than setting up a separate server as a hot spare, since the responder runs on Equalizer. However, while Responders are capable of returning only a single HTML page, a hot spare can be configured to return multiple HTML pages and images. See "Adding a Responder" on page 616for information on configuring a responder.

To use a hot spare, you would usually configure it on Equalizer as follows:

1. Set **Maximum Reused Connections** to zero (0), so that all connection requests sent to the hot spare are accepted.
2. Enable the **Hot Spare** flag. This specifies that any requests refused by all the other server instances in a server pool because they reached their **Maximum Reused Connections** limit (or are down) will be forwarded to the hot spare server.
3. Enable the **Dont Persist** flag so that connections made to the hot spare don’t persist. Each connection to the cluster must first be load balanced amongst the other servers in the cluster and only go to the hot spare if all the other servers have reached their **Maximum Reused Connections** limit.

Setting Initial Weights for Mixed Clusters

Equalizer enables you to build heterogeneous clusters using servers of widely varying capabilities. Adjust for the differences by assigning initial weights that correspond to the relative capabilities of the available servers. This enables you to get the most out of existing hardware, so you can use an older server side-by-side with a new one.

After you assign relative initial weights, monitor cluster performance for two to three hours under load. You will probably fine-tune the weights and optimize performance of your cluster two or three times.

Continue monitoring the performance of your cluster and servers and watch for any trends. For example, if you notice that Equalizer *always* adjusts the dynamic weights so that the weight of one server is far below 100 and the weight of another is far above 100, the server whose dynamic weight is consistently being reduced might have a problem.

Interaction of Server Options and Connection Processing

Server option settings have a direct influence on connection and request processing, particularly Layer 4 and Layer 7 persistence. (Note that persistence is set at the cluster level, but can be disabled for individual servers using the **Dont Persist** option.) The hierarchy of server option settings is shown in the table below:

Server Option	Description
Server disabled	An **initial weight** of **0** tells Equalizer that no traffic should be sent to the server, disabling the server. This option setting takes precedence over all other options (including persistence, hot spare, etc.).
Max Connections (> 0)	If set to a non-zero value, Equalizer limits the total number of simultaneously open connections to the server to that value. This limit is not overridden if the **Hot Spare** option is enabled on a server, and is not overridden by a Layer 4 sticky record or Layer 7 persistence cookie for the server in an incoming request.
Quiesce Enabled	The server is *not* included in load balancing decisions, so that no *new* connections will be made to this server. If a request in an incoming connection has an existing Layer 4 sticky record or Layer 7 cookie for a server, however, the request will be sent to that server even when **Quiesce** is enabled. If **dont persist** is also enabled on the server, the sticky record or cookie is ignored.
Hot Spare Enabled	The server is *not* included in load balancing decisions, so that traffic is sent to this server *only* when no other server in the cluster is available to accept client connections. If a request in an incoming connection has an existing Layer 4 sticky record or Layer 7 cookie for a server, however, the request will be sent to that server even when **hot spare** is enabled. If **dont persist** is also enabled on the server, the sticky record or cookie is ignored.

Shutting Down a Server Gracefully

To avoid interrupting user sessions, make sure that a server to be shut down or deleted from a cluster no longer has any active connections. When a server’s initial weight is zero, Equalizer will not send new requests to that server. Connections that are already established continue to exist until the client and server application end them or they time out because they are idle.

To shut down servers in a generic TCP or UDP (L4) cluster, you can set the server’s weight to zero and wait for the existing connections to terminate. However, you need to quiesce servers in HTTP and HTTPS (L7) clusters to enable servers to finish processing requests for clients that have a persistent session with the server.

When you quiesce a server, Equalizer does not route new connections from new clients to the server, but will still send requests from clients with a persistent session to the quiescing server. Once all the persistent sessions on the server have expired, you can set the server’s initial weight to zero; then Equalizer will not send additional requests to the server.

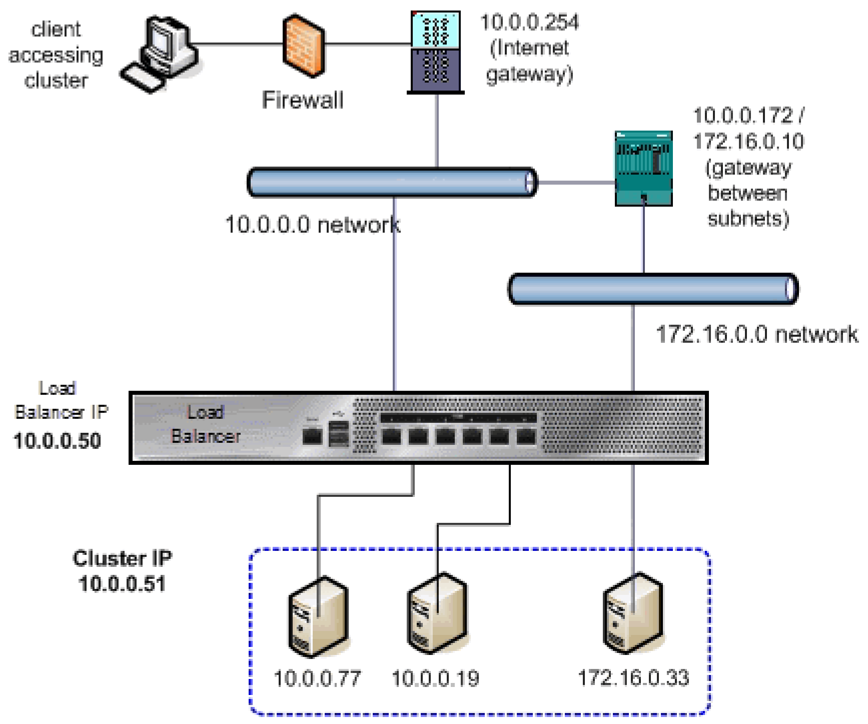
Note that while a server instance is quiescing, it will still receive new requests *if all of the other server instances in a server pool are unavailable*. This behavior prevents any new requests from being refused, but may lengthen the time needed to terminate all active persistent connections.

Server Configuration Constraints

When configuring servers on Equalizer, you must observe the following constraints:

- In general, there must be no Layer 3 devices (e.g., such as a router) between a server and Equalizer in order for health check probes to work correctly.
- Equalizer operation depends on reliable communication between Equalizer and the servers behind it. The latency introduced by Layer 3 devices such as routers can, for example, result in connection time-outs even when the server is available.
- If you require remote servers in your clusters, you must be very careful to configure network routes that allow Equalizer and the server to communicate. It may also be necessary to reconfigure probe time-outs in order to account for network latency.

An example of a configuration with both directly connected servers and remotely accessible servers is illustrated in the diagram below.



The configuration shown above is an example of a single VLAN configuration, where Equalizer communicates with all servers and clients via the same subnet. The example cluster shown above contains three servers, two on the local 10.0.0.0 subnet, and one on another subnet.

In this example, a static route would be needed on Equalizer to forward all packets for the 172.16.0.0 network to the gateway at 10.0.0.172. Similarly, the server at 172.16.0.33 would need a static route that forwards all traffic for the 10.0.0.0 network through 172.16.0.10, the gateway address for the 10.0.0.0 network.

Configuring Routing on Servers

The way you configure routing on servers behind Equalizer depends largely on whether Equalizer's **spoof** option is enabled on a cluster.

Spoof Controls SNAT

If **spoof** is *disabled*, SNAT (Source Network Address Translation) is performed on client requests before sending them on to the server -- the source address used in the packet sent to the server is Equalizer's IP address on the VLAN used to communicate with the server.

If **spoof** is *enabled*, SNAT is *not* performed on client requests before sending them on to the server -- the source address used in the packet sent to the server is the *client's* IP address.

How Spoof Influences Routing

When **spoof** is *disabled*, special routing is usually not required on servers, since they will respond to Equalizer's IP address on the appropriate VLAN.

When **spoof** is *enabled*, you should configure your servers so that Equalizer gateways the packets the servers send to clients. If you do not adjust the routing on your servers when the **spoof** option is enabled, servers will not route responses through Equalizer and clients receiving such responses directly from servers will drop the responses and the client connection will time out. An easy way to do this is to configure the server's default gateway to be an address on an Equalizer subnet. If this is not possible, then static routes should be used to properly route client requests back to Equalizer.

Direct Server Return (DSR) configurations with Layer 4 clusters are an exception to this rule. In DSR configurations, client requests coming through Equalizer are routed to servers, which then respond directly back to the clients without going through Equalizer. Therefore, servers in a DSR configuration typically have a default gateway other than Equalizer.

In non-DSR clusters with **spoof** enabled, you should use one of the following Equalizer addresses as the default gateway on the server (for the server instance on the server pool in the cluster):

- **If the servers are connected to a single (standalone) Equalizer**, the default gateway IP address that you should use on the server is Equalizer's IP address on the VLAN associated with the Equalizer front-panel port to which the server is connected.
- **If the servers are connected to two Equalizers in a failover configuration**, the default gateway IP address that you should use on the server is always Equalizer's failover IP address on the VLAN associated with the Equalizer front-panel port to which the server is connected.

The commands or utilities that you use to configure routing on a server depends on the server's operating system, but usually involves some form of the route command. Check your server operating system documentation. To verify that you have configured a server's routing correctly, trace the route from the server to a destination address outside the internal network to ensure that Equalizer gets used as a gateway. On UNIX systems, use the traceroute utility; on Windows, use tracert.

Note that you should configure routing on each server from the server's system console, not through a telnet session. This will avoid any disconnects that might otherwise occur as you adjust the network settings on the server.

Server Statistics and Reporting

The CLI display of Statistics can be seen by entering the following within the server context:

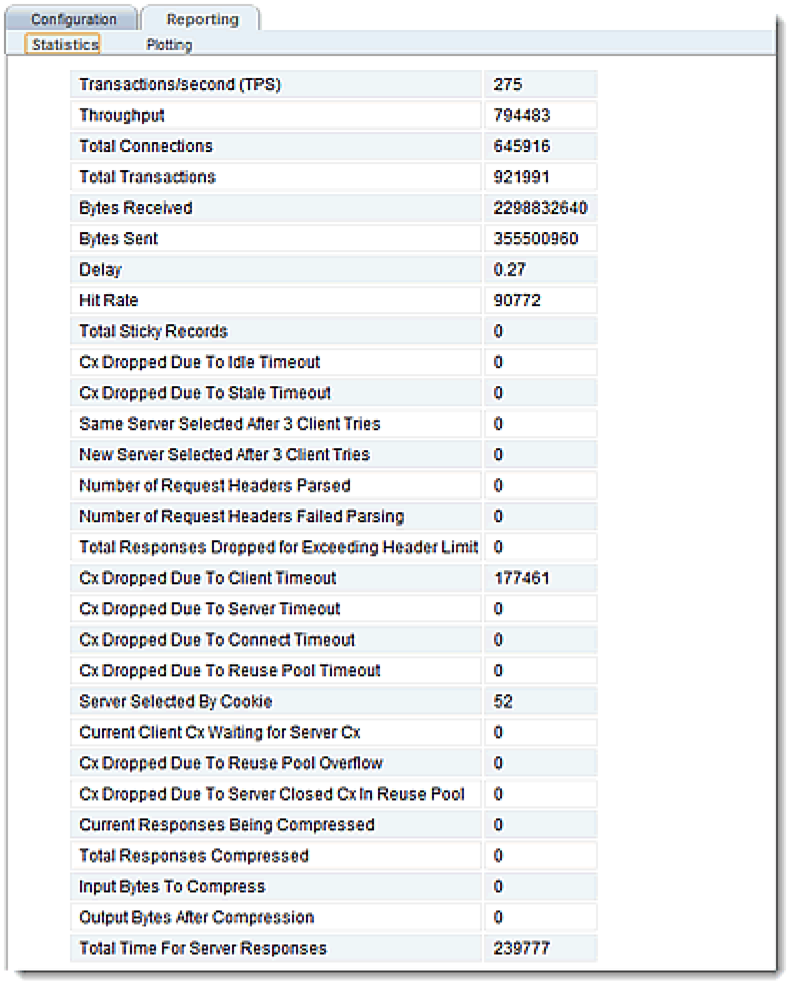
Sample Server Statistics Display (CLI)

```
eqcli sv-spi*> stats
                    Current     60 sec      10 min      60 min
TOTALPRCSD          133250      0           0           0
TOTALRESPPRCSD      281157      0           0           0
TIMESPENT           146728992   N/A         N/A         N/A
ACTIVECONX          0           0           0           0
BYTERCVD            4251815936  N/A         N/A         N/A
BYTESEND            72129144    N/A         N/A         N/A
TOTALSTKY           0           N/A         N/A         N/A
CURRSTKY            0           0           0           0
IDLECONXDROPED      0           N/A         N/A         N/A
STALECONXDROPED     0           N/A         N/A         N/A
FAILTHRICE          0           N/A         N/A         N/A
NEWFAILTHRICE       0           N/A         N/A         N/A
RSPPARSED           281157      N/A         N/A         N/A
RSPFAILED           0           N/A         N/A         N/A
RSPFAILHDR          0           N/A         N/A         N/A
CLNTTO              734         N/A         N/A         N/A
SRVRTO              41182       N/A         N/A         N/A
CONNTO              63477       N/A         N/A         N/A
SELPERSIST          0           N/A         N/A         N/A
SPLICE              69772       N/A         N/A         N/A
CURSRVERUSE         0           0           0           0
CURCLNTWAITQ        0           0           0           0
REUSEOF             0           N/A         N/A         N/A
REUSESRVR           0           N/A         N/A         N/A
REUSETO             0           N/A         N/A         N/A
CURCOMP             0           0           0           0
TOTALCOMP           0           N/A         N/A         N/A
eqcli sv-spi*>
```

To view the GUI display:

1. Verify that you are logged into the GUI. (Refer to "Logging In" on page 238.)
2. Select the **Load Balance** configuration tab on the left navigational pane if it is not already selected.
3. Click on the arrow (▶) beside **Servers**to expand the branch.
4. Select a Server and click on the **Reporting** tab to display statistics. The following is an example of the statistics displayed.

Sample Server Statistics GUI Display (GUI)



Transactions/second (TPS)	275
Throughput	794483
Total Connections	645916
Total Transactions	921991
Bytes Received	2298832640
Bytes Sent	355500960
Delay	0.27
Hit Rate	90772
Total Sticky Records	0
Cx Dropped Due To Idle Timeout	0
Cx Dropped Due To Stale Timeout	0
Same Server Selected After 3 Client Tries	0
New Server Selected After 3 Client Tries	0
Number of Request Headers Parsed	0
Number of Request Headers Failed Parsing	0
Total Responses Dropped for Exceeding Header Limit	0
Cx Dropped Due To Client Timeout	177461
Cx Dropped Due To Server Timeout	0
Cx Dropped Due To Connect Timeout	0
Cx Dropped Due To Reuse Pool Timeout	0
Server Selected By Cookie	52
Current Client Cx Waiting for Server Cx	0
Cx Dropped Due To Reuse Pool Overflow	0
Cx Dropped Due To Server Closed Cx In Reuse Pool	0
Current Responses Being Compressed	0
Total Responses Compressed	0
Input Bytes To Compress	0
Output Bytes After Compression	0
Total Time For Server Responses	239777

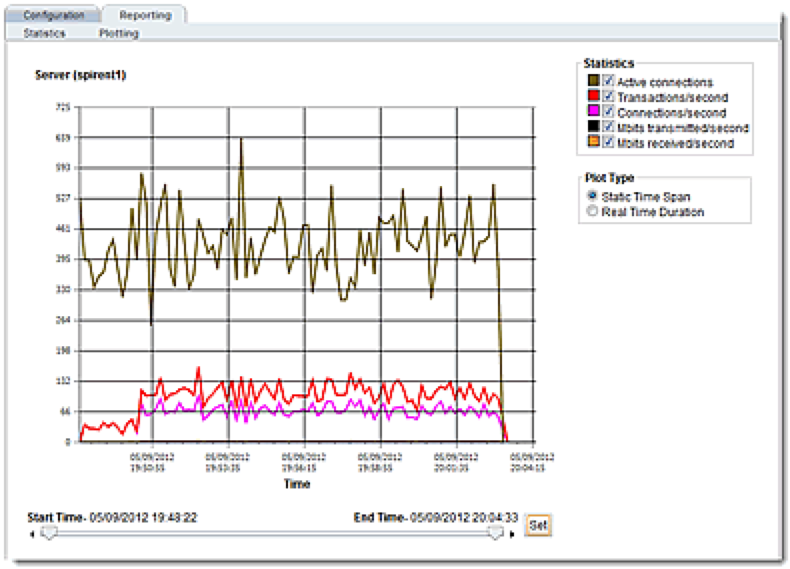
Server Statistic Definitions

CLI Term	GUI Term	Definition
TOTALPRCSD	N/A	Connections processed.
TOTALRESPPRCSD	Total Transactions	Responses processed.
TIMESPENT	Total Time For Server Responses	The total time spent on this object.
ACTIVECONX	Active Connections	Active connections.
BYTERCVD	Bytes Received	Bytes received.
BYTESEND	Bytes Sent	Bytes transmitted.
TOTALSTKY	Total Sticky Records	Total sticky connections.
CURRSTKY	Current Sticky Records	Current sticky records.
IDLECONXDROPED	Cx Dropped Due To Idle Timeout	Connections dropped for idle timeout.
STALECONXDROPED	Cx Dropped Due To Stale Timeout	Connections dropped for stale timeout.
FAILTHRICE	Same Server Selected After 3 Client Tries	Same server selected after 3 retries.
NEWFAILTHRICE	New Server Selected After 3 Client Tries	New server selected after 3 retries.
RSPPARSED	Number of Request Headers Parsed	Response headers parsed.
RSPFAILED	Number of Request Headers Failed Parsing	Responses failed header parsing.
RSPFAILHDR	Total Responses Dropped for Exceeding Header Limit	Responses dropped for exceeding the header limit.
CLNTTO	Cx Dropped Due To Client Timeout	Connections dropped due to client timeout.
SRVRTO	Cx Dropped Due To Server Timeout	Connections dropped due to server timeout.
CONNTO	Cx Dropped Due To Connect Timeout	Connections dropped due to connect timeout.
SELPERSIST	Cx Dropped Due To Reuse Pool Timeout	The number of times the server was selected by a cookie.
SPLICE	Server Selected By Cookie	The total number of client-server connections linked.
CURSRVERUSE	Server Cx In Reuse Pool	The number of connections in the TCP MUX reuse pool.
CURCLNTWAITQ	Client Cx In Wait Queue	The number of client connections in the wait queue.

CLI Term	GUI Term	Definition
REUSEOF	Cx Dropped Due To Reuse Pool Overflow	The number of connections dropped due to reuse pool overflow.
REUSESRVR	Cx Dropped Due To Server Closed Cx In Reuse Pool	The number of connections dropped due to server closing a connection in the reuse pool.
REUSETO	Current Client Cx Waiting for Server Cx	Total connections timed out in TCP MUX reuse pool.
CURCOMP	Current Responses Being Compressed	Current responses being compressed.
TOTALCOMP	Total Responses Compressed	Total responses compressed.
N/A	Input Bytes To Compress	Input Bytes To Compress
N/A	Output Bytes After Compression	Output Bytes After Compression

The following is a graphical plot that can be displayed on the GUI. Select a server e on the left navigational pane and click on the **Reporting** tab and then **Plotting**. The following will be displayed:

Sample Server Plot



Statistics
- 🟫 ✓ Active connections
- 🟥 ✓ Transactions/second
- 🟪 ✓ Connections/second
- ⬛ ✓ Mbits transmitted/second
- 🟧 ✓ Mbits received/second

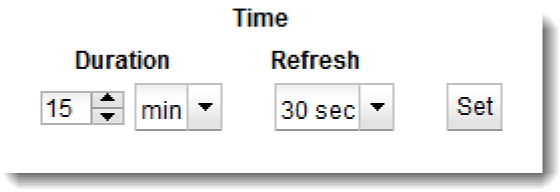
Plot Type
- ● Static Time Span
- ○ Real Time Duration

Start Time- 05/09/2012 19:43:22

End Time- 05/09/2012 20:04:33

The specific types of statistics that are displayed are determined by the selections on the **Statistics** pane on the upper right corner of the GUI.Make selections based on the data that you require.

The **Plot Type** selection determines whether the display shown reflects a Static Time Span which is configured using the slider or whether a real time duration is display. If **Real Time Duration** is selected the slider controls will change to **Duration** and **Refresh** controls as shown below. In this case set the **Duration** of time in which you would like to review statistics and the **Refresh** rate desired.



Duration: 15 min

Refresh: 30 sec

Chapter 16

Health Checks

Sections within this chapter include:

Overview of Health Checks

Health Checks are a series of timed probes sent by Equalizer to connected objects. The probe responses influence load balancing decisions and determine the pool of possible load balancing targets. The responses can be in either of two forms: They can simply let Equalizer know whether a server is UP or DOWN. These are called *liveness* checks. They can also be in the form of a returned load value from a server agent or VMware's status reporting. These are called *load* checks.

Liveness checks are health checks attached that can be attached to servers, server pools, server instances, and Link Load Balancing (LLB) Gateways. If, based on attached liveness checks, an object is determined to be DOWN, it is removed from the pool of possible load balancing targets.

Load checks evaluate returned probing results to determine load on a server instance. Using server instance delay, active connections, individual load check loads, and server pool average load check loads, a single load value is produced for each server instance. This load value is then utilized to influence load balancing decisions.

Health checks are configurable load balancing objects on Equalizer. Each health check features its own set of options, parameters, and timers that are used in the probing process. After configuration, the health checks are used by "attaching" then to servers, server pools, server instances, or LLB gateways. When attached, the configured health checks will probe the target objects (to which they are attached) using the configured parameters and timers.

When the health checks are attached to objects they are enabled by default, and can be disabled if necessary.

Health checks are configured using various protocols. The following are the health checks supported by Equalizer:

Layer 3 ICMP Health Checks

Layer 3 ICMP health are liveness checks that involve a "ping" to the server IP address to determine whether or not the host is UP.

Layer 4 TCP/UDP Health Checks

Layer 4 TCP/UDP health checks are liveness checks. At Layer 4, Equalizer attempts to connect to a specific TCP or UDP port. A TCP SYN request to port 80, for example, would be sent and checks for a TCP SYN ACK in return. If the response is not received within the configured timeout intervals, port 80 would be marked DOWN for that server.

Active Content Verification (ACV) Health Checks

ACV health checks are liveness checks. They are a mechanism for checking the validity of servers, server pools, and server instances. When you attach an ACV health check to an object, Equalizer requests data from the object and verifies that the returned data contains a character string that indicates that the data is valid. You can use ACV with most network services that support a text-based request/response protocol, such as HTTP. ACV health checks can be used with TCP clusters only.

Server Agent Health Checks

Server Agent health checks are load checks. In Server Agent health checking, the load on a target is measured by a user-supplied "agent", running on the target server. The server responds to health check queries with load "values", describing it's current level of load. Since the "agent" runs on the target, it has access to a wealth of information used to determine the system's load, thus providing an accurate picture of the target's load.

VLB Health Checks

VLB health checks are load checks. They are used when you have one or more VMware ESX Servers. You can configure health checks to use VMware’s status reporting to determine server, and server instance status.

About this Chapter

This chapter features:

- A discussion on the different types of health checks and how they are used in load balancing decisions.
- Definitions of key terms used throughout the chapter.
- Health check constraints.
- Ports used for health check probes.
- How health checks are displayed in both the CLI and the GUI.
- A discussion of the System Health Checks.
- Firmware upgrade notes.
- Instructions for creating health checks, attaching them to the load balancing objects, disabling , deleting, and configuring them.
- A discussion of the health check timeouts.

Key Terms

The following terms will be used extensively throughout this chapter. It is recommended that you become familiar with the terminology prior to configuring any of the health checks described.

Key Term	Definition
Load Balancing Object	A general term that can refer to a server, server instance, server pool, or LLB Gateway.
Liveness Check	A **Liveness Check** is a health check that reports or is utilized only to assess UP/DOWN status. L3 (Ping), L4 (UDP and TCP), and ACV are liveness checks.
Load	Load is a measure of activity level, used for making load balancing decisions. The term **Load** is used in the following contexts: **Health Check** -- The **Load** on associated with a load check is the value reported in the response to the health check query. There is no load associated with a liveness check. A liveness check reflects only UP/DOWN status. **Server Instance** -- The **Load** on a server instance is calculated using 3 types of measure: delay, active connections, and health check load(s).
Load Check	A **Load Check** is a health check that reports a load value, and may also (directly or indirectly) indicate UP/DOWN status. VLB and Server Agent are types of **Load Checks**.
ACV	Active Content Verification. Used to check the validity of a load balancing object based on the validity of returned character strings
ICMP	Layer 3 ICMP Probes.
TCP	Layer 4 TCP Probes.
UDP	Layer 4 UDP Probes.
srvagt	Server Agent Probes.
VLB	VLB Health Checks.
Attached Health Check	The health checks that are attached to load balancing objects.
Target Object Parameter (also access_parms)	Can either be 'Self', indicating that the parameters required to access the object to be health-checked are in the health check definition, or "Parent", indicating that the parameters are taken from the object to which the health check is attached if not available in the health check definition.
LLB Gateway	Refers to Link Load Balancing Gateway.

Health Check Constraints

The following table shows the load balancing objects and the health checks that can be attached to each:

Load Balancing Objects	Types of health check that can be attached
Servers **Server Instances**	Liveness Checks (UP/DOWN)--ONLY. These health checks can be attached to a server, or server instance. In addition, Equalizer will enforce the following restrictions: •a Layer 4 TCP health check can only be attached to a TCP-type server (or server instance) •a Layer 4 UDP health check can only be attached to a UDP type server (or server instance) Server Agent and VLB Health Checks CANNOT be used with these objects.
Server Pools	Any health check can be attached to a server pool. All server instances in the pool are probed as specified by the health check attached to the server pool. Equalizer will enforce the following restrictions on combinations that do not make sense: •You cannot add an L4 TCP health check to a server pool of UDP servers •You cannot add an L4 UDP health check to a server pool of TCP servers •You cannot add a VLB health check to a server pool of mixed virtual machines and real servers
LLB Gateways	ICMP Health Checks only.

Health Check Probing Ports

The table below summarizes the health checking probes and ports used with Equalizer:

Layer	Protocol	Port	Details
Layer 3	ICMP	N/A	Echo request/reply
Layer 4	UDP/IP	53	DNS
		111	(RPC4) Portmap
		2049	(RPC4) NFS
Layer 4	TCP/IP	User supplied	TCP (connect only)
Layer 7		User supplied	ACV (Plaintext)
		Use supplied	ACV (SSL)
		1510 (default)	Server Agent Health Check
		N/A	VLB Health Check

System Health Checks

There are three pre-defined System Health checks that are automatically attached to load balancing objects:

- **ICMP-Default** - attached to servers when they are created.
- **TCP-Default** - attached to a TCP server pools when the first TCP server instance is added.
- **UDP-Default** - attached to a UDP server pool when the first UDP server instance is added.

System Health Check Notes

1. The System Health Checks are assigned to new objects as described above. Instances of the health checks can be added to servers, server pools, and server instances.
2. Whenever a server instance is attached to server pool without existing server instances, a TCP or UDP default health check is automatically assigned to the server pool, based on the server protocol.
    - If you delete a system health check attached to a server pool, a new TCP or UDP system health check will be automatically attached to the server pool when you add the first server instance to the server pool. This will occur even if you deleted a previously existing system health check.
    - If you delete a system health check attached to a server pool and custom health checks remain attached, a new TCP or UDP system health check will be automatically attached to the server pool when you add the first server instance. This will occur even if you deleted a previously existing system health check. The existing custom health checks will be unaffected.
3. The System Health Checks cannot be deleted.
4. The System Health Checks can be disabled at a global level or individually, on an attached instance.
5. The **IP**(`ip`), **Port** (`port`) (TCP and UDP) and **Use Parent Parameters** (`access_parm`) parameters on the default health checks are NOT modifiable.
6. When you upgrade from a release prior to EQ/OS 10.3.2 the health checks will be configured identically with the L3 and L4 Health Checks from previous releases. In previous EQ/OS releases, the system automatically enabled an L3 (ICMP) health check on a server on creation and an L4 TCP health check on server instances on creation.

Firmware Upgrade Notes

When upgrading or downgrading, it is HIGHLY RECOMMENDED that you create a backup of your current configuration. Refer to "Backup and Restore" on page 286 **for instructions on creating backups and restoring previous configurations.**

For Layer 4 TCP and ACV Health Checks Upgrading to EQ/OS 10.3.2 or higher

In previous OS releases, custom L4 TCP/ACV probe parameters could be configured on a Server Pool. When your system is upgraded to EQ/OS 10.3.2 or higher, the non-default probe parameters will be copied to the new health check configuration IF AND ONLY IF the '"Probe Layer 4"' option is ENABLED on the Server Instance in the Server Pool (in the older configuration). If this flag is DISABLED in the older configuration, the non-default parameters will be lost and will revert to the default values on any Layer 4 TCP health checks added to that Server Pool on the upgraded system.

For Layer 4 UDP Health Checks Upgrading to EQ/OS 10.3.2 or higher

If there are UDP health checks attached to server instances when your system is upgraded to EQ/OS 10. the health checks will copied to the new health check configuration IF AND ONLY IF the '"Probe Layer 4"' option is ENABLED on the Server Instance in the Server Pool (in the older configuration). If this flag is DISABLED in the older configuration, the health check WILL NOT be copied.

If all of the *server instances* in a server pool have the same UDP health check attached with the "Probe Layer 4" option enabled, a new UDP health check will be attached to the *server pool* when upgrading. This means that the health check will apply to all server instances.

For Server Agent and VLB Health Checks Upgrading to EQ/OS 10.3.2 or higher

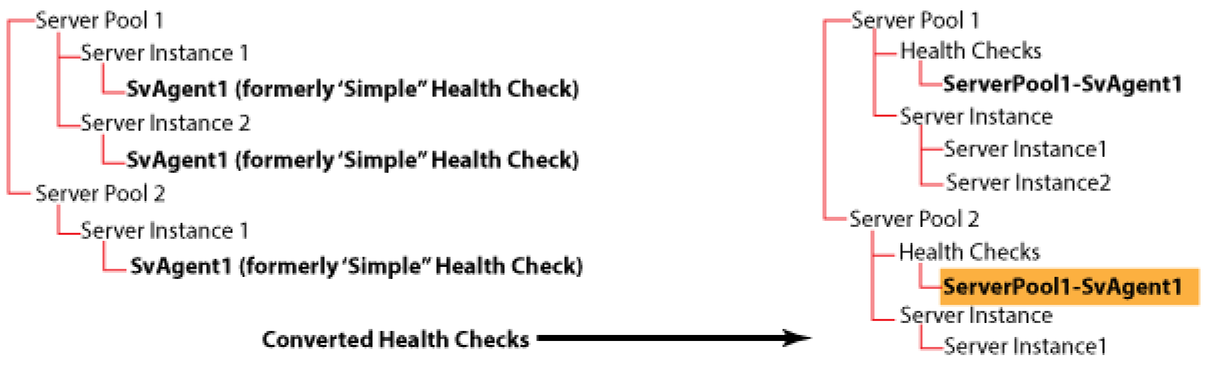
If a VLB or Server Agent health check was *not* previously attached to *each* of the server instances in a server pool, a new VLB or Server Agent health check object will be created when upgrading. The new health check objects will maintain the same parameters as your previous configuration, however, they will not be attached to the server instances as they were before upgrading. You then have the option of attaching the newly created health check object to Server Pools or Server Instances.

If a VLB or Server Agent health check was previously attached to *each* server instance in a server pool, the health checks will be attached to the Server Pool when upgrading. They will maintain the same parameters as your previous configuration. In this case, the health checks will apply to all server instances in the server pool.

For Server Agent and VLB Health Checks upgrading to EQ/OS 10.3.2b or higher

If you are upgrading from a firmware version prior to EQ/OS 10.3.2b and VLB or Server Agent health checks with the same parameters are attached to server instances prior to upgrade, the same health checks will be generated with an appended name and attached to the server pools in the new configuration.

The figure below shows the health check conversion during upgrade. In the example, a Server Agent health check is used. The process is the same for VLB health checks.



1. All Server Agent or VLB Health checks having the same configuration and were previously attached to server instances as "health check instances" in the older configuration will be converted to load balancing objects in the new configuration. They can be displayed in the CLI by entering `eqcli > **show health_check**` or in the GUI by selecting **Load Balance > Health Check** on the left navigational pane.
2. The health checks will be attached to the server pools and not the server instances they were attached to previously, however, they will be renamed as shown in the figure above. The naming convention used is:
   `SERVERPOOLNAME-HEALTHCHECKNAME`
3. Note that the health check attached to **ServerPool2 > ServerInstance3** above (**ServerPool1-SvAgent1**) uses the same naming convention, even though it is attached to **ServerPool 2**. You cannot "rename" the health check. If this name is unacceptable, you can create a new health check (See "Health Check Configuration Process" on page 557) using the same parameters as the converted health check and attach it to server instances as necessary. Remove the converted health check from the server pool as well.

Health Check Configuration Process

The health check configuration process uses the same basic steps for all health check types. These include:

1. Creating the health check.
2. Configuring health check parameters using the CLI or GUI.
3. Attaching health checks to load balancing objects (servers, server pools, server instances, and LLB gateways).

All health checks are enabled, by default. They can be disabled either globally or for a single, attached health check as necessary. Refer to "Disabling Health Checks" on page 603 for instructions.

Configuring ICMP Health Checks

ICMP (Layer 3 Internet Control Message Protocol) health check probes are "liveness" checks. This means that if a response is not received to an ICMP echo request, and no other health check probes are configured, the object is marked DOWN. However, Equalizer will continue to send ICMP requests to the IP address and, if an ICMP echo response is subsequently received, the object will be marked UP.

The number of times that a server is probed before being marked DOWN is determined by the **Maximum Tries per Interval** parameter. The **ICMP Probe Interval** parameter is a timer and when it starts, the first ICMP probe is sent. If the value of **Maximum Tries per Interval** is greater than 1, the next probe is sent a number of seconds later equal to:

```
        (ICMP Probe Interval) / (ICMP Probe Maximum Tries)
```

For example, the default **Probe Interval** is "15" and the default **Maximum Tries per Interval** is "3". So, by default, an ICMP probe is sent every 5 seconds.

When the **Probe Interval** timer expires, an object is marked UP if a response to any probe sent during the **Probe Interval** was received. An object is marked DOWN by lack of a response to an ICMP probe during the probe interval. (Refer to the description of the **Relaxed** probe interval in the table below)

The ICMP-Default health will be attached to servers when they are created.

ICMP health checks can be disabled either globally or by attached health checks using either the CLI or the GUI. Refer to "Disabling Health Checks" on page 603 for procedures.

Parameters

ICMP probes are configured using the parameters described in the ICMP Health Check Probe Parameters table below.

Further descriptions about the health check timers and intervals and how they work together are provided in "Health Check Timeouts" on page 610.

ICMP Health Check Probe Parameters

GUI Parameters (CLI Parameter)	Description
Target Object Parameters (Cannot be modified on ICMP-Default Health Checks)	
Use Parent Parameters (access_parms)	If this option is set to **parent**, this indicates that any access parameters needed for the health check probe that are not available in the health check definition will be taken from the object to which the health check is attached. If this option is set to **self**, and some or all of the the required probe parameters are missing, then the health check will be marked as incomplete until parameters are entered.
Target Object IP (ip)	The IP address of the health check target.
Health Check Timers	
Max Tries per Interval (probe_maxtries)	The maximum number of times per **Probe Interval** that Equalizer will attempt to probe the object.(default: **3**) It is recommended that this parameter be set to 3 or greater in configurations where a physical server appears in multiple server pools.
Probe Interval (probe_interval)	A timer specifying the length of time (in seconds) during which a successful server probe must occur, or the server is marked DOWN. At least one L3 ICMP probe for the server must have succeeded since the last Equalizer reboot, or failed ICMP probes for the server will be ignored and the server will be marked UP.(default: **15**)
Flags	
Disable (disable)	Disables the health check on a global level.
Relaxed (relaxed_probe)	When enabled, if a server is DOWN, but has not previously been UP, it will be marked UP. Enabling this option prevents the sudden reporting of servers being DOWN following an upgrade.

Configuring an ICMP Health Check Using the CLI

1. Log in to the CLI.
2. Create a new ICMP health check using the following format:

```
eqcli > health_check healthchecknametype icmpparameter_namevalue [...]
```

   where:

   healthcheckname - is the name of the health check.

   parameter_name and ***value*** - is the health check parameter and the value assigned to it.

   For example:

```
eqcli > health_check icmp_test type icmp ip 1.3.5.7
eqcli: 12000287: Operation successful

eqcli > show health_check icmp_test
Health Check Name     : icmp_test
Type                  : icmp
Access Parameter      : parent
IP                    : 1.3.5.7
Probe Interval        : 15
Max Tries/Interval    : 3
Flags                 :
eqcli >
```

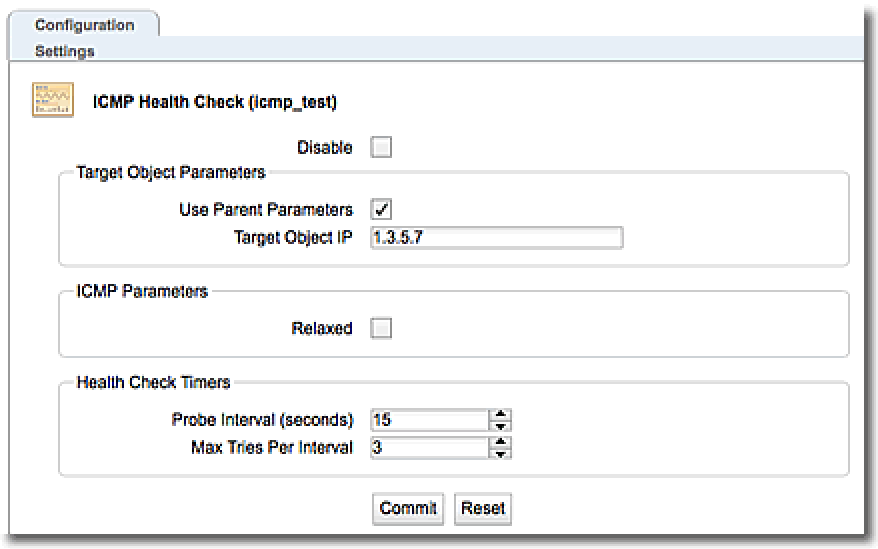
3. Enter additional **parameter_names** and **values** described in the ICMP Health Check Probe Parameters table above. Use the following format:

Additional information on using the health check CLI commands can be found in "Health Check Commands" on page 189.

Configuring an ICMP Health Check Using the GUI

If not using the ICMP-Default health check, you can create an ICMP health check as follows:

1. Log in to the GUI.
2. Click on the **Load Balance** configuration tab.
3. Do either of the following to display the **Add Health Check** form:
   a. Right click on the **Health Checks** branch and select **Add Health Check.**
   b. Select the **Health Checks** branch on the left navigational pane to display the list of configured health checks on the right configuration pane. Click on the + icon.
4. Enter a name in the **Health Check Name** box and select **ICMP** from the drop down list.
5. Click on the **Commit** button to save the health check. The **Configuration > Settings** screen will appear on the right configuration pane.



ICMP Health Check (icmp_test)

Disable	☐

Target Object Parameters

Use Parent Parameters	✓
Target Object IP	1.3.5.7

ICMP Parameters

Relaxed	☐

Health Check Timers

Probe Interval (seconds)	15
Max Tries Per Interval	3

4. Modify the ICMP health check probe parameter shown in the ICMP Health Check Probe Parameters table above as necessary. Clicking on the **Reset** button will remove anything that you've entered and return the screen to the default display.
5. Click on **Commit**.

Refer to "Attaching Health Checks to Load Balancing Objects" on page 592 for instructions on attaching the health check.

Configuring TCP & UDP Health Checks

Layer 4 TCP and UDP health checks are "liveness" checks. At Layer 4, Equalizer sends health check probes and attempts to connect to a specific TCP or UDP port. A TCP SYN request to port 80, for example, would expect a TCP SYN ACK response in return. If the response is not received within the configured timeouts and intervals, port 80 would be marked DOWN for that server, or server instance.

L4 UDP probes are performed on UDP protocol servers and are valid for IPv4 addresses only.

For specific Remote Procedure Call (RPC) services running on well-known ports - Network File System (NFS) and portmap - an RPC call is sent to the server. If no response is received the server is marked DOWN.

For the Domain Name System (DNS), a DNS request is sent to the server. If no response is received the server is marked DOWN.

For all other UDP services, a UDP datagram is sent to the server probe port and if no response is received the server is marked DOWN.

Layer 4 TCP and UDP health checks can be disabled either globally or by attached health check using either the CLI or the GUI. Refer to "Disabling Health Checks" on page 603 for procedures.

When you configure a server pool with TCP server instances. The server pools will have one TCP-Default health check. (See "System Health Checks" on page 554).Likewise, when you configure a server pool using UDP server instances, a UDP-Default health check will be attached to the server instance.

UDP Probing Behavior

UDP probes are sent via ports 53, 111, and 2049. On all other ports, a “dummy” probe, which would not be recognizable to a receiving service, is also transmitted.

- If a UDP response is received, the probe is considered UP.
- If an ICMP "destination unreachable" response is received for either port or protocol, the probe is considered “DOWN”.
- If no response is received, the UP/DOWN determination depends on the health check’s **Relax** (`relaxed_probe`in the CLI) option is enabled.
  - If the **Relax** (`relaxed_probe`) option is enabled, the probe is considered “UP”.
  - If the **Relax** (`relaxed_probe`) option is *not* enabled, the probe is considered “DOWN”.
- In all cases, the state of a probe or the object to which it is attached will not change until one probe interval has passed since the probe began. This prevents spurious UP/DOWN indications and alerts.
- UDP probing is only valid for IPv4 addresses.

Parameters

The table below describes the TCP and UDP health check parameters used in the configuration process.

Further descriptions about the health check timeouts and intervals and how they work together are provided in "Health Check Timeouts" on page 610.

L4 Health Check Parameters

GUI Parameter (CLI Parameter)	Description
Target Object Parameters (Cannot be modified on TCP-Default Health Checks)	
Use Parent Parameters **(access_parms)**	If this option is set to **parent**, this indicates that any access parameters needed for the health check probe that are not available in the health check definition will be taken from the object to which the health check is attached. If this option is set to **self**, and some or all of the required probe parameters are missing, then the health check will be marked as incomplete unless parameters are entered.
Target Object IP **(ip)**	The IP address of the health check target.
Protocol **(proto)** **(For UDP only)**	This drop down list specifies that the Layer 4 health check protocol. (default: **DNS**)
Port **(port)**	This is the port to use for the port number for probing the server.
SSL Parameters (For TCP only)	
Highest TLS Version **(highest_tls_version)**	Used when the **Use SSL (probe_ssl)** option is enabled. Sets the health check highest TLS version. It specifies the highest TLS version that will be offered in the SSL probe sent to servers The probe can use levels from **SSLv3** and the highest probe levels of **TLS v1.0, v1.1**, or **v1.2**. (default:**TLS v1.2**) In the CLI, values are used to assign the TLS versions as follows: 1 = SSL v3 2 = TLS v1 3 = TLS v1.1 4 = TLS v1.2 (Default: **4**)
Health Check Timers	
Probe Interval (seconds) **(probe_interval)**	A timer specifying the length of time (in seconds) during which a successful server probe must occur, or the server is marked DOWN. (default: **15**)

GUI Parameter (CLI Parameter)	Description
Max Tries Per Interval **(`probe_maxtries`)**	The maximum number of times per **Probe Interval** that Equalizer will attempt to probe an object.(default: **3**) It is recommended that this parameter be set to 3 or greater in configurations where a physical server appears in multiple server pools.
Probe Global Timeout (seconds) **(`probe_gto`)**	The health check probe global timeout (in seconds). (default: **5**)
Probe Connect Timeout (seconds) **(`probe_cto`)** **(For TCP only)**	The health check connection timeout. The number of seconds (default: **1**) that Equalizer will wait for a connection attempt to the health check server application to succeed before marking the object UP or DOWN.
Probe Data Timeout (seconds) **(`probe_dto`)** **(For TCP only)**	The health check data timeout. Once a connection is established, this parameter indicates the number of seconds (default: **2**) Equalizer will wait for the first byte of the health check server application response before marking the object DOWN.
Flags	
Disable **(`disable`)**	Disables the health check on a global level.
Use SSL **(`probe_ssl`)** **(ACV, Server Agent, and TCP only)**	If enabled, the probe exchange between Equalizer and a server instance will be performed over an encrypted SSL connection. The **Highest Version** drop down list will display a list of the highest TLS version to use. (default: **Enabled**)
Relaxed **(`relaxed_probe`)** **(UDP only)**	When this option is enabled, if the probe receives either no response or an ICMP response other than port or protocol unreachable, the probe will be marked UP. ICMP responses of port or protocol unreachable always cause the probe to be considered down.

Configuring TCP and UDP Health Checks Using the CLI

If not using the TCP-Default or UDP-Default health check, you can create one as follows:

1. Log in to the CLI.
2. Create a new L4 health check using the following format:

```
eqcli > health_check healthcheckname type [tcp|udp] parameter_name value [...]
```

where:

healthcheckname - is the name of the health check.

parameter_name and ***value*** - is the health check parameter and the value assigned to it. Refer to the L4 Health Check Parameters table above.

For example:

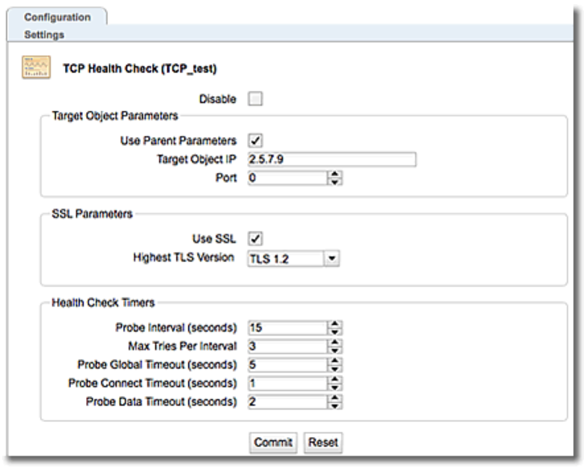
```
eqcli > health_check TCP_test type tcp ip 2.5.7.9 probe_maxtries 2
eqcli: 12000287: Operation successful

eqcli > show health_check TCP_test

Health Check Name     : TCP_test
Type                  : tcp
Access Parameter      : parent
IP                    : 2.5.7.9
Port                  : 0
Probe Interval        : 15
Max Tries/Interval    : 2
Global Timeout        : 5
Connection Timeout    : 1
Data Timeout          : 2
Highest TLS Version   : 4
Flags                 : probe_ssl
eqcli >
```

Additional information on using the health check CLI commands can be found in "Health Check Commands" on page 189.

Configuring TCP and UDP Health Checks Using the GUI

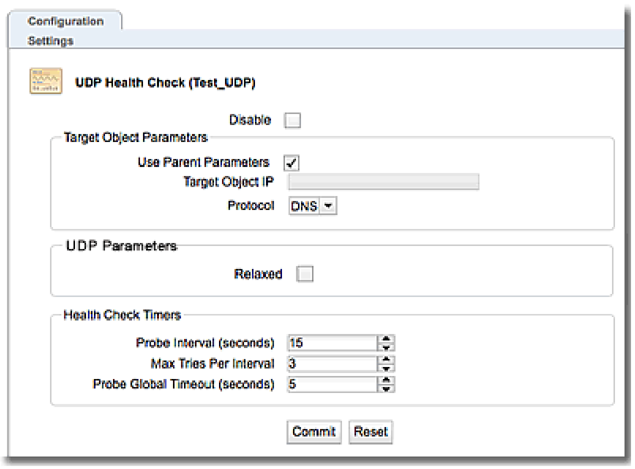
If not using the TCP-Default or UDP-Default health check, you can create one as follows:

1. Log in to the GUI.
2. Click on the **Load Balance** configuration tab.
3. Do either of the following to display the **Add Health Check** form:
   a. Right click on the **Health Checks** branch and select **Add Health Check.**
   b. Select the **Health Checks** branch on the left navigational pane to display the list of configured health checks on the right configuration pane. Click on the **+** icon.
4. Enter a name in the **Health Check Name** box and select **UDP** or **TCP**from the drop down list.
5. Click on the **Commit** button to save the health check. The **Configuration > Settings** screen will appear on the right configuration pane. The TCP health check and UDP health check configuration screens are shown.



TCP Health Check (TCP_test)

Disable	☐
Target Object Parameters	
Use Parent Parameters	✓
Target Object IP	2.5.7.9
Port	0
SSL Parameters	
Use SSL	✓
Highest TLS Version	TLS 1.2
Health Check Timers	
Probe Interval (seconds)	15
Max Tries Per Interval	3
Probe Global Timeout (seconds)	5
Probe Connect Timeout (seconds)	1
Probe Data Timeout (seconds)	2



UDP Health Check (Test_UDP)

Disable	☐
Target Object Parameters	
Use Parent Parameters	✓
Target Object IP	
Protocol	DNS
UDP Parameters	
Relaxed	☐
Health Check Timers	
Probe Interval (seconds)	15
Max Tries Per Interval	3
Probe Global Timeout (seconds)	5

6. Modify the parameters as shown in the L4 Health Check Parameters and Configuring TCP & UDP Health Checks tables above as necessary.
7. Click on **Commit.**

Refer to "Attaching Health Checks to Load Balancing Objects" on page 592 for instructions on attaching the health check. In addition attached Layer 4 health checks can check the UP/DOWN status of the selected server, or servers, where server instances are attached. Refer to "Attached ACV and TCP Health Checks and the Test Function" on page 596 for details.

Configuring ACV Health Checks

Active Content Verification (ACV) is a "liveness" health check that serves two purposes: Layer 4 probing and Layer 7 probing. It is a mechanism used to check the validity of a server instance. When an ACV health check is attached to a server instance (in a server pool), Equalizer requests data from the server and verifies that the returned data contains a character string that indicates that the data is valid. If the returned data matches the character string configured in the ACV health check, the server is considered to be UP. If the character string is not found, the server is marked DOWN.

An ACV query string can be specified if the service running on the server's probe port requires input in order to respond. If the TCP probe connection is not established ACV probing will fail as well.

You can use ACV with most network services that support a text-based request/response protocol, such as HTTP.

ACV can be used with Layer 4 TCP clusters and NOT Layer 4 UDP clusters

ACV probes are limited to 99 characters, and must use only the printable ASCII characters (decimal 32 to 126). Equalizer searches for ACV response strings in the first 1024 characters of the server's response to high-level TCP probes. If an ACV response string is not found, the server is marked DOWN.

ACV is best explained using a simple example. HTTP protocol enables you to establish a connection to a server, request a file, and read the result. The example below shows the connection process when a user requests a telnet connection to an HTTP server and requests an HTML page.

```
> telnet www.myserver.com 80    >>>>>>>>>>>>>     User requests connection to server.
Connected to www.myserver.com   >>>>>>>>>>>>>     Telnet indicates connection is established.
> GET /index.html       >>>>>>>>>>>>>>>>>>>>>     User sends request for HTML page.
<HTML>  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>     Server responds with requested page.
<TITLE>Welcome to our Home Page </TITLE>
</HTML>
Connection closed by foreign host       >>>>     Telnet indicates server connection closed.
```

Equalizer can perform the same exchange automatically and verify the server pool's response by checking the returned data against an expected result.

ACV Query and Response Strings

Specifying an *ACV Query* and an *ACV Response String* basically automates the exchange shown above. Equalizer uses the probe string to request data from each server. To verify the server's content, it searches the returned data for the response string. For example, you can use "**`GET /index.html`**" as the *ACV Query* and you can set the *Response String* to some text, such as "Welcome" in the example shown above, which appears on the home page.

Similarly, if you have a web server with a PHP application that accesses a database, you can use ACV to ensure that all of the components in the application are working. You could set up a PHP page called **test.php** that accesses the database and returns a page containing **ALL OK** if there are no problems.

For most applications, only an ACV response string is needed - Equalizer connects to the probe port on the server instance and waits for a response.

Some applications may require input on the connection before a response is sent back to Equalizer. The ACV query string is used for this purpose - if it is non-empty, the ACV query string is sent after the server instance connection is established.

An ACV query or response string:

- Must be enclosed in single or double quotes if it contains a space character.
- Any single or double quotes included within the string must be preceded by the backslash character (\).

Note -In ACV Query strings character escapes such as "\n" for new-line, "\r" for carriage return and "\t" for Tab are supported. "\r" and "\n" must be manually inserted at the end of all HTTP and HTTPS ACV probes. For ACV Response strings Regular Expression matching is supported.

Attached ACV health checks can run a "test" on query and response strings configured in the health check to check the UP/DOWN status of the selected server, or servers, where server instances are configured in the server pool. Refer to "Attached ACV and TCP Health Checks and the Test Function" on page 596 for instructions on using this feature.

ACV health checks can be disabled either globally by attached health check using either the CLI or the GUI. Refer to "Disabling Health Checks" on page 603 for procedures.

Parameters

The table below describes the ACV health check parameters that are used in the configuration process.

Further descriptions about the health check timers and intervals and how they work together are provided in "Health Check Timeouts" on page 610.

ACV Health Check Parameters

GUI Parameter (CLI Parameter)	Description
Target Object Parameters	
Use Parent Parameters (`access_parms`)	If this option is set to **parent**, this indicates that any access parameters needed for the health check probe that are not available in the health check definition will be taken from the object to which the health check is attached. If this option is set to **self**, and some or all of the the required probe parameters are missing, then the health check will be marked as incomplete unless parameters are entered.
Target Object IP (`ip`)	The IP address of the health check target.
Port (`port`)	This is the port to use for the port number for probing the server.
ACV Parameters	
ACV Query (`query`)	For most applications, only an ACV response string is needed - It connects to the probe port on the server instance and waits for a response. Some applications may require input on the connection before a response is sent back. The **ACV query** string is used for this purpose - if it is non-empty, the ACV query string is sent after the server instance connection is established. An ACV query or response string: Must be enclosed in single or double quotes if it contains a space character. Any single or double quotes included within the string must be preceded by the backslash character (\). Character escapes such as “\n” for new-line, “\r” for carriage return and “\t” for Tab are supported. "\r" and "\n" must be manually inserted at the end of all HTTP and HTTPS ACV probes.For ACV Response strings Regular Expression matching is supported.

GUI Parameter (CLI Parameter)	Description
ACV Response **(response)**	The response string should be text that appears only in a valid response. This string is also case-sensitive. An example of a poorly chosen string would be HTML, since most web servers automatically generate error pages that contain valid HTML. If the page that is returned contains the correct response string (in the first 1024 characters, including headers) the server is marked "UP"; if ALL OK were not present, the server is marked "DOWN".
SSL Parameters	
Highest Version **(highest_tls_version)**	Used when the **Use SSL (probe_ssl)** option is enabled. Sets the health check highest TLS version. It specifies the highest TLS version that will be offered in the SSL probe sent to servers The probe can use levels from SSLv3 and the highest probe levels of **TLS v1.0, v1.1**, or **v1.2**. (default:**TLS v1.2**) In the CLI, values are used to assign the TLS versions as follows: 1 = SSL v3 2 = TLS v1 3 = TLS v1.1 4 = TLS v1.2 (Default: 4)
Health Check Timers	
Probe Interval (seconds) **(probe_interval)**	A timer specifying the length of time (in seconds) during which a successful server probe must occur, or the server is marked "down". (default: **15**)
Max Tries Per Interval **(probe_maxtries)**	The maximum number of times per **Probe Interval** that Equalizer will attempt to probe an object.(default: **3**) It is recommended that this parameter be set to 3 or greater in configurations where a physical server appears in multiple server pools.
Probe Global Timeout (seconds) **(probe_gto)**	The health check probe global timeout (in seconds). (default: **5**)
Probe Connect Timeout (seconds) **(probe_cto)**	The health check connection timeout. The number of seconds (default: **1**) that Equalizer will wait for a connection attempt to the health check server application to succeed before marking the object UP or DOWN.
Probe Data Timeout (seconds) **(probe_dto)**	The health check data timeout. Once a connection is established, this parameter indicates the number of seconds (default: **2**) Equalizer will wait for the first byte of the health check server application response before marking the object DOWN.

GUI Parameter (CLI Parameter)	Description
Flags	
Disable **(`disable`)**	Disables the health check on a global level.
Use SSL **(`probe_ssl`)**	If enabled, the TCP probe exchange between Equalizer and a server instance will be performed over an encrypted SSL connection. The **Highest Version** drop down list will display a list of the highest TLS version to use. Enabled by default.

Configuring ACV Health Checks using the CLI

Configure an ACV health check using the CLI as follows:.

1. Log in to the CLI.
2. Create a new ACV health check using the following format:

```
eqcli > health_check healthcheckname type acv parameter_name value [...]response
responsestring query
querystring flags flag
```

where:

healthcheckname - is the name of the health check.

parameter_name and ***value*** - is the health check parameter and the value assigned to it. Refer to the ACV Health Check Parameters table above.

responsestring - is the response string for the health check.[1]

querystring - is the query string for the health probes[2].

For example:

```
eqcli > health_check ACV_test type acv ip 5.6.7.9 response !#@&^&8 query
!#@&^&8

eqcli > show health_check ACV_test

Health Check Name     : ACV_test
Type                  : acv
Access Parameter      : parent
IP                    : 5.6.7.9
Port                  : 0
Probe Interval        : 15
Max Tries/Interval    : 3
Global Timeout        : 5
Connection Timeout    : 1
Data Timeout          : 2
Highest TLS Version   : 4
Query                 : !#@&^&8
Response              : !#@&^&8
Flags                 : probe_ssl
eqcli >
```

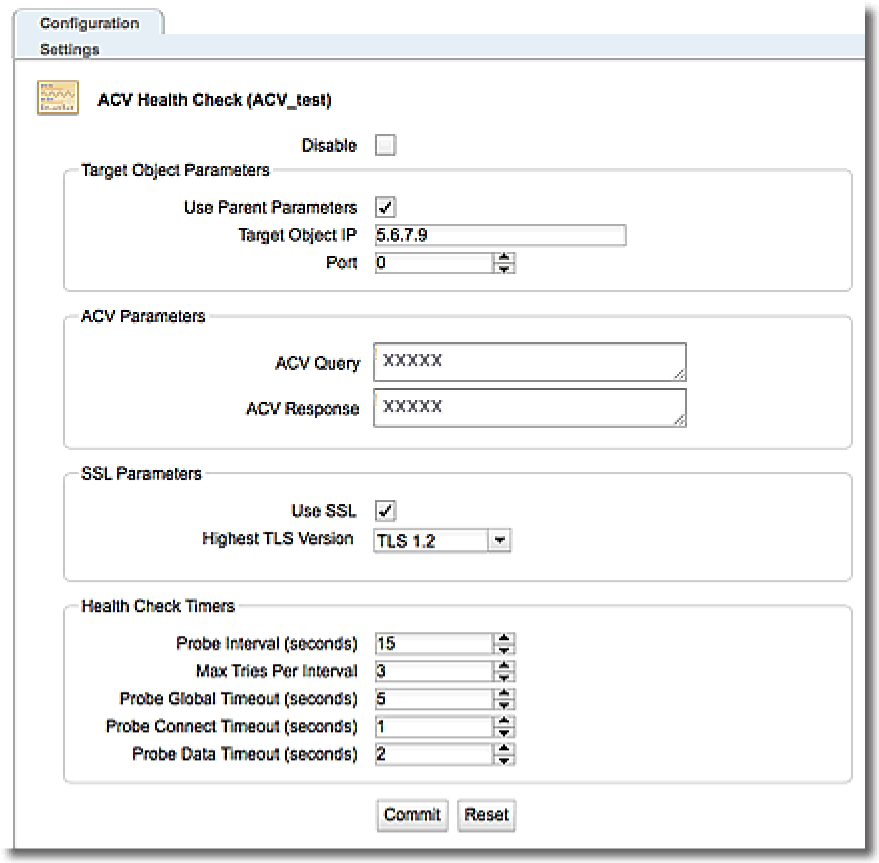
Additional information on using the health check CLI commands can be found in"Health Check Commands" on page 189.

[1]The responsestring shown below is for demonstration purposes only.
[2]The querystring shown below is for demonstration purposes only.

Configuring ACV Health Checks using the GUI

Configure an ACV health check using the GUI as follows:

1. Log in to the GUI.
2. Click on the **Load Balance** configuration tab.
3. Do either of the following to display the **Add Health Check** form:
   a. Right click on the **Health Checks** branch and select **Add Health Check.**
   b. Select the **Health Checks** branch on the left navigational pane to display the list of configured health checks on the right configuration pane. Click on the + icon.
4. Enter a name in the **Health Check Name** box and select a **ACV**from the drop down list.
5. Click on the **Commit** button to save the health check. The **Configuration > Settings** screen will appear on the right configuration pane.[1]



ACV Health Check (ACV_test)

Disable	☐

Target Object Parameters

Use Parent Parameters	✓
Target Object IP	5.6.7.9
Port	0

ACV Parameters

ACV Query	XXXXX
ACV Response	XXXXX

SSL Parameters

Use SSL	✓
Highest TLS Version	TLS 1.2

Health Check Timers

Probe Interval (seconds)	15
Max Tries Per Interval	3
Probe Global Timeout (seconds)	5
Probe Connect Timeout (seconds)	1
Probe Data Timeout (seconds)	2

[1]The ACV Query and ACV Response strings shown in the screen capture below is for demonstration purposes only.

6. Configure the ACV health check using the parameters in the ACV Health Check Parameters table above. Clicking on the **Reset** button will remove anything that you've entered and return the screen to the default display. Clicking on the **Reset** button will remove anything that you've entered and return the screen to the default display.
7. Click on the **Commit** button to save the health check.

Refer to "Attaching Health Checks to Load Balancing Objects" on page 592 for instructions on attaching the health check.

Configuring VLB Health Checks

All Equalizers support basic load balancing of VMware servers through VMware vConsole integration. Equalizer uses VMware's management API to retrieve real-time virtual server performance information from a VMware vCenter console that manages virtual machines running on ESX Server (or from a single ESX Server directly). The additional server availability and resource utilization information obtained from VMware allows Equalizer to more efficiently direct the traffic flowing to VMware virtual machines.

VLB health checks can be configured on server pools only.

Messages will appear in the Equalizer log (on the global **Logging > Event Log** tab) when Equalizer communicates with VMware, and when the state of a VM server changes. Otherwise, VLB works behind the scenes to provide accurate and detailed VM server status information that Equalizer uses to make well-informed load balancing decisions.

VLB health checks can be disabled either globally or by attached health check using either the CLI or the GUI. Refer to"Disabling Health Checks" on page 603 for procedures.

The basic process for configuring a VLB health check is as follows:

1. Create a VLB Manager as an External Service.
2. Associate a server with a Virtual Machine on VMware.
3. Create the health check
4. Configure VLB health check parameters.
5. Attach the health check.

Parameters

The table below describes the VLB health check parameters that are used in the configuration process.

Further descriptions about the health check timers and intervals and how they work together are provided in "Health Check Timeouts" on page 610.

VLB Health Check Parameters

GUI Parameter (CLI Parameter)	Description
Target Object Parameters	
Use Parent Parameters **(access_parms)**	If this option is set to **parent**, this indicates that any access parameters needed for the health check probe that are not available in the health check definition will be taken from the object to which the health check is attached. If this option is set to **self**, and some or all of the the required probe parameters are missing, then the health check will be marked as incomplete unless parameters are entered.
VLB Manager **(vlb_mgr)**	Sets the VLB Manager for the health check.
UUID **(vlb_uuid)**	Universally Unique Identifier (UUID) Sets the VLB UUID for a target server.
VM Resource to Check	
VLB Parameter **(vlb_param)**	Options are vm_cpu or vm_ram. **vm_cpu:** Selects the CPU load value for monitoring **vm_ram:** Selects the RAM usage value for monitoring.
Health Check Timers	
Probe Interval **(probe_interval)**	The number of seconds (default: **15**) Equalizer will wait for a health check attempt to succeed before marking a server down.
Max Tries per Interval **(probe_maxtries)**	The maximum number of health check connection attempts per probe interval before marking a server down. (default: **3**) It is recommended that this parameter be set to 3 or greater in configurations where a physical server appears in multiple server pools.
Flags	
Disable **(disable)**	Disables the health check on a global level.

Configuring VLB Health Checks Using the CLI

Configure a VLB health check using the CLI as follows:

Create a VLB Manager as an External Service

1. Log in to the CLI.
2. A VLB Manager is a saved configuration by which Equalizer communicates with VMware. Enter the following a the `eqcli` command prompt to create a VLB Manager as an External Service:

```
eqcli > ext_services vlb_manager <name>
```

where:

name - is the name of the vlb manager

3. Enter the new VLB Manager, adding a **`URL`**, **`username`**, **`password`**, **`Connect Timeout`** parameters and flags. Enter:

```
eqcli xs vlb-nam* > URL value
eqcli xs vlb-nam* > username name
eqcli xs vlb-nam* > password name
eqcli xs vlb-nam* > flags disable[a]
```

a. The only flag used is **`disable`** which would disable the VLB Manager if necessary.

4. Enter the following to verify the new VLB Manager and parameters. In the example below, a VLB manager "`esxi-01`" is created.

```
eqcli > ext_services vlb_manager esxi-01
eqcli xs-vlb-esx*> show
VLB Manager Name : esxi-01
URL : https://192.168.213.196/sdk
Username : root
Timeout : 0
Flags :
```

Note - For security reasons, a Password value is not displayed.

Associate a Equalizer server with a Virtual Machine on VMware

5. Show the configured server by entering the server context and then entering:

```
eqcli > server servvername show
```

where

servername - is the name of the server. An example with a server "`centos216`" is shown below.

```
This server is enabled.
Server Name : centos216
IP Address : 192.168.213.216
Port : 22
Protocol : tcp
VID : 1
Max Reuse Connections : 0
Reuse Connections Timeout : 0
VLB Manager :
UUID :
Flags : probe_l3
```

6. Enter the server context and set the `vlb_manager` value by entering the following. In this example the `vlb_manager` is "`esxi-01`" on a server "`centos216`":

```
eqcli sv-cen*> vlb_manager esxi-01
eqcli sv-cen*> commit
eqcli: 12000287: Operation successful
```

Create a Health Check object

7. The next step is to create a VLB health check. Use the following format:

```
   eqcli > health_check healthchecknametype vlb
```

   where:

   healthcheckname - is the name of the health check

   For example:

```
eqcli > health_check VLB_Test type vlb

eqcli > show health_check VLB_test
Health Check Name     : VLB_Test
Type                  : vlb
Access Parameter      : parent
VLB Manager           : esxi-01
VM UUID               :
VLB Parameter         : vm_cpu
Probe Interval        : 15
Max Tries/Interval    : 3
Flags                 :
eqcli >
```

   By default, the **access_parm** parameter will be enabled so that he parameters of the VM will be inherited.

Configure additional health check parameters

8. Configure additional VLB health check parameters using the "Configuring VLB Health Checks" on page 576 table above as reference. Refer to "Health Check Commands" on page 189 for additional information on entering parameters. Use the format:

```
eqcli > health_check healthcheckname parameter_name value [...]
```

   where:

   healthcheckname - is the name of the health check.

   parameter_name and ***value [...]*** are health check parameters that can be found on the "Configuring VLB Health Checks" on page 576 table above.

Attach the health check to a server pool

9. Attach the VLB health check to a server pool. Refer to "Attaching Health Checks to Load Balancing Objects" on page 592"Attaching Health Checks to Load Balancing Objects" on page 592for instructions.

Additional information on using the health check CLI commands can be found in "Health Check Commands" on page 189.

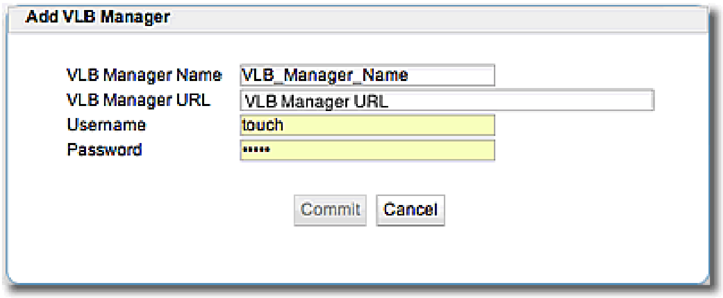
Configuring VLB Health Checks Using the GUI

Configure a VLB health check using the GUI as follows:

Create a VLB Manager as an External Service

A VLB Manager is a saved configuration by which Equalizer communicates with VMware.

1. Click on + on the right configuration pane to add a VLB Manager.The figure below will be displayed. The screen features accordion panes for the existing and the VLB managers that are labeled. Clicking on the delete icon will delete the health check whose accordion pane is currently open.
2. Log in to the GUI.
3. Select the **System > External Services > VLB Manager** on the left navigational pane.



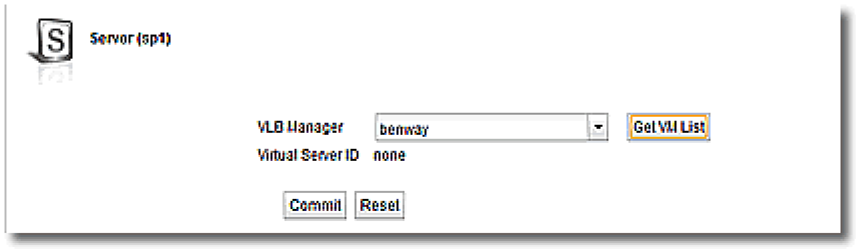
Add VLB Manager

VLB Manager Name	VLB_Manager_Name
VLB Manager URL	VLB Manager URL
Username	touch
Password	•••••

4. Enter a URL for the VLB Manager you would like to connect with in the **VLB Manager URL** field. Add **Username**/**Password** credentials for login as well.
5. Click on **Commit** to continue.

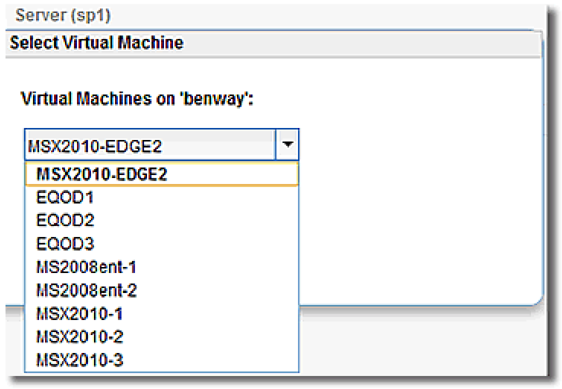
Associate a Server with a Virtual Machine on VMware

6. Associate an Equalizer server with a Virtual Machine on VMware by selecting the desired Equalizer Server on the left navigational pane and then selecting **Configuration > VLB** to display the figure below.


Server (sp1)

VLB Manager benway
Virtual Server ID none

The **VLB Manager** drop-down list lists all the VLB managers defined in the External Services context. [Default **None**] Select a **VLB Manager** from the drop-down list above and click **Get VMList**. The figure below will be displayed.


Server (sp1)
Select Virtual Machine

Virtual Machines on 'benway':

MSX2010-EDGE2
MSX2010-EDGE2
EQOD1
EQOD2
EQOD3
MS2008ent-1
MS2008ent-2
MSX2010-1
MSX2010-2
MSX2010-3

The pop up contains the list of the Virtual Machines (VMs) retrieved from the VLB Manager. The VM with the matching IP address (if found) is pre-chosen (highlighted) in the list. Click on **Select** to select the pre-highlighted VM, or choose another before clicking **Select**.

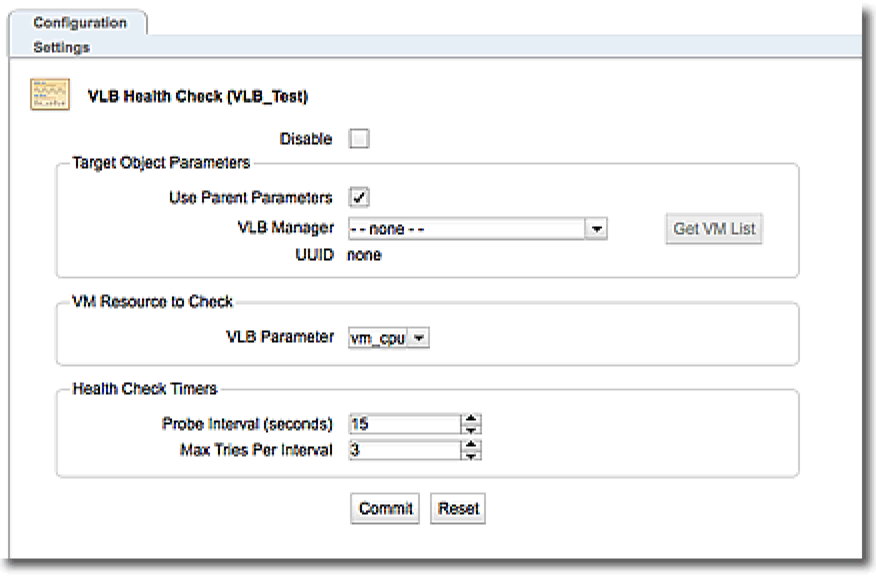
The tab is then redisplayed with the **Virtual Server ID** of the selected VM.

7. Click on **Commit** to save the setting.

Note - If getting the VM list from the VLB Manager fails, an Error popup is displayed with the message: "Failed to associate virtual server on VLB manager <name>." plus the detailed message returned from VMware.

Create the Health Check

8. Log in to the GUI.
9. Click on the **Load Balance** configuration tab.
10. Do either of the following to display the **Add Health Check** form:
    a. Right click on the **Health Checks** branch and select **Add Health Check.**
    b. Select the **Health Checks** branch on the left navigational pane to display the list of configured health checks on the right configuration pane. Click on the **+** icon.
11. Enter a name in the **Health Check Name** box and select **VLB**from the drop down list.
12. Click on the **Commit** button to save the health check. The **Configuration > Settings** screen will appear on the right configuration pane.



VLB Health Check (VLB_Test)

Disable ☐

Target Object Parameters

Use Parent Parameters ✓

VLB Manager -- none --

UUID none

VM Resource to Check

VLB Parameter vm_cpu

Health Check Timers

Probe Interval (seconds) 15

Max Tries Per Interval 3

By default, the **Use Parent Parameters** option will be enabled so that he parameters of the VM will be inherited

Select a VLB Manager and UUID for the Health Check

13. Click on the VLB health check on the left navigational pane to display the configuration screen on the right.
14. Click on the **VLB Manager** drop down list and select a VLB Manager to use for this health check. The associated UUID will be displayed when a selection is made. Clicking on the **Reset** button will remove anything that you've entered and return the screen to the default display
15. Click on **Commit**. to save the health check.

Configure additional VLB health check parameters

16. Configure additional VLB parameters using the VLB Health Check Parameters table above as reference.

Attach the health check to a server pool

17. Attach the VLB health check to a server pool. "Attaching Health Checks to Load Balancing Objects" on page 592 for instructions.

Configuring Server Agent Health Checks

A Server Agent health check is a "load check" health check that provides a load measurement (response) to indicate a server's UP/DOWN status. It is used with server pools.

A user-supplied "server agent" must be running at the target, which supplies a load value in response to a query from Equalizer. This information is obtained by the server agent by any means available at the target server. For example, if supported by the target, the server agent program could check current CPU or memory allocation to determine a load value.The only requirement from Equalizer's perspective is that the server agent's response must be in the form of a single integer or floating point value. The server agent may either return the load value immediately upon accepting a connection on the configured port, or it may require a stimulus string before returning the value. After returning the value, the server agent closes the connection and waits for another connection.

Server Agent health checks can be disabled either globally or by attached health check using either the CLI or the GUI. Refer to "Disabling Health Checks" on page 603 for procedures.

Server Agents

A server agent is a custom written application that runs on a server and listens on a specific port (default: 1510). When a connection request is received on that port, the server agent returns an integer value between -1 and 100 that indicates the relative load on the server (-1 meaning the server should be considered unavailable, 0 meaning very lightly loaded, and 100 meaning heavily loaded). Server agents can be used with any cluster type, and have an effect on all load balancing policies except round robin, and static, which ignore server agent return values.

Agent Probe Process

When Equalizer connects to the port on which a server agent is running, it uses the number returned by the agent in its load balancing calculations, with the server agent policy giving highest preference to the server agent’s return value over other factors.

The number returned by the agent to Equalizer is intended to indicate the current load on the server. The agent application that runs on the server can be written in any available scripting or programming language and can use any appropriate method to determine server load.

Server agents should be running on all server instances in the server pool. However, by default, a server is not marked DOWN when an agent value is not returned and the attached health check instance flag is set to **Optional** .

Sample Server Agent

You can create custom Server Agents as shell scripts, or in Java, Perl, C, or other languages. The code snippet below is an example of a simple server agent example written in Perl. This code assumes that an integer response value is supplied on the command line and returns that value when a connection is made on port 1510 (configurable via the server instance **Probe Port** (`probe_port`) variable).

This sample agent is intended for testing purposes only. In a real deployment, the server agent would determine the response value to return by polling system resources, or some other real-time method.

```
#!/usr/bin/perl -w
# serveragent.pl
#--------------------
#(c) Copyright 2014 Fortinet, Inc.

use strict;
use Socket;

# use port 1510 as default
my $port = 1510;
my $proto = getprotobyname('tcp');

# take the server agent response value from the command line
my $value = shift;
my $response = "$value\n";

# response has to be a valid server agent response
$response==-1 or ($response > 0 and $response<101)
or die "Response must be between -1 and 100";

# create a socket and set the options, set up listen port
socket(SERVER, PF_INET, SOCK_STREAM, $proto) or die "socket: $!";
setsockopt(SERVER, SOL_SOCKET, SO_REUSEADDR, 1) or die "setsock: $!";
my $paddr = sockaddr_in($port, INADDR_ANY);

# bind to the port, then listen on it
bind(SERVER, $paddr) or die "bind: $!";
listen(SERVER, SOMAXCONN) or die "listen: $!";
print "Server agent started on port $port\n";

# accepting a connection
my $client_addr;
while ($client_addr = accept(CLIENT, SERVER)) {

# find out who connected
my ($client_port, $client_ip) = sockaddr_in($client_addr);
my $client_ipnum = inet_ntoa($client_ip);


# print who has connected -- this is for debugging only
  print "Connection from: [$client_ipnum]\n";

# send the server agent response value
  print CLIENT $response;

# close connection
  close CLIENT;
}
```

Here is the output of the server program when it is started on the server:

```
$ ./serveragent.pl 50
Server agent started on port 1510
Connection from: [10.0.0.32]
```

Another “Connection” line prints each time the server agent is probed by Equalizer.

From Equalizer’s perspective, the only value returned by the server agent is the integer set on the command line. For example, if you use the example server agent above and set the response to “50”, here is what you will see if you use the **telnet** command to open the server agent IP and port:

```
$ telnet 10.0.0.120 1510
50
Connection to host lost.
```

Parameters

Note - Server Agent health checks work with load balancing policies other than **round robin** and **static**. **Round robin**and **static** ignore any agent response for all server instances in a server pool. All other policies use the integer returned by the agent as one factor in determining the server to which a new request is sent.

The table below describes the Server Agent health check parameters used in the configuration process.

Further descriptions about the health check timers and intervals and how they work together are provided in "Health Check Timeouts" on page 610.

Server Agent Health Check Parameters

GUI Parameter (CLI Parameter)	Description
Target Object Parameters	
Use Parent Parameters **(access_parms)**	If this option is set to **parent**, this indicates that any access parameters needed for the health check probe that are not available in the health check definition will be taken from the object to which the health check is attached. If this option is set to **self**, and some or all of the the required probe parameters are missing, then the health check will be marked as incomplete unless parameters are entered.
Target Object IP **(ip)**	This is the IP address of the health check target.
Port **(port)**	This is the port to use for the port number for probing the server.
Server Agent Parameters	

Server Agent Health Check Parameters

GUI Parameter (CLI Parameter)	Description
Lightest Load Value (`light_load`)	A floating point value that is the 'healthiest' (or least busy) load value that can be returned by the health check server application. For example: if the application returns a value of -1 to indicate that it is DOWN, then set **light_load** to -1. (default: **0.000000**)
Heaviest Load Value (`heavy_load`)	A floating point value that is the busiest (or most highly loaded) load value that can be returned by the health check server application for the health check. For example: if the application returns a value of 100 to indicate that it is very heavily loaded, then set **heavy_load** to 100. (default: **100.000000**)
Health Check Query (`query`)	A string sent to the server agent after a connection is established. For example: a server agent health check application may require a string such as get load \r \n before it will send a load value. This parameter is optional.
SSL Parameters	
Highest Version (`highest_tls_version`)	Used when the **Use SSL (probe_ssl)** option is enabled. Sets the health check highest TLS version. It specifies the highest TLS version that will be offered in the SSL probe sent to servers The probe can use levels from SSLv3 and the highest probe levels of **TLS v1.0, v1.1**, or **v1.2**. (default: **TLS v1.2**) In the CLI, values are used to assign the TLS versions as follows: 1 = SSL v3 2 = TLS v1 3 = TLS v1.1 4 = TLS v1.2 (Default: 4)
Health Check Timers	
Probe Interval (`probe_interval`)	The number of seconds (default: **15**) Equalizer will wait for a health check attempt to succeed before marking a server down.
Max Tries Per Interval (`probe_maxtries`)	The maximum number of health check connection attempts per probe interval before marking a server down. (default: **3**) It is recommended that this parameter be set to 3 or greater in configurations where a physical server appears in multiple server pools.
Probe Global Timeout (`probe_gto`)	The health check global timeout. The number of seconds (default: **5**) Equalizer waits for a connection to the health check server application to complete before marking the server down.

Server Agent Health Check Parameters

GUI Parameter (CLI Parameter)	**Description**
Probe Connect Timeout (probe_cto)	The health check connection timeout. The number of seconds (default: **1**) that Equalizer will wait for a connection attempt to the health check server application to succeed before marking the server down.
Probe Data Timeout (probe_dto)	The health check data timeout. Once a connection is established, this parameter indicates the number of seconds (default: **2**) Equalizer will wait for the first byte of the health check server application response before marking the server down.
Flags	
Disable (disable)	Disables the health check on a global level.
Use SSL (probe_ssl)	If enabled, the probe exchange between Equalizer and a server instance will be performed over an encrypted SSL connection. The **Highest Version** drop down list will display a list of the highest TLS version to use. (default: **Enabled**)

Configuring Server Agent Health Checks Using the CLI

1. Log in to the CLI.
2. Create a new Server Agegnt health check using the the following format:

```
eqcli > health_check healthcheckname type srvagt parameter_namevalue [...]
```

where:

healthcheckname - is the name of the health check.

parameter_name and ***value*** - is the health check parameter and the value assigned to it. Refer to Server Agent Health Check Parameters table above.

For example:

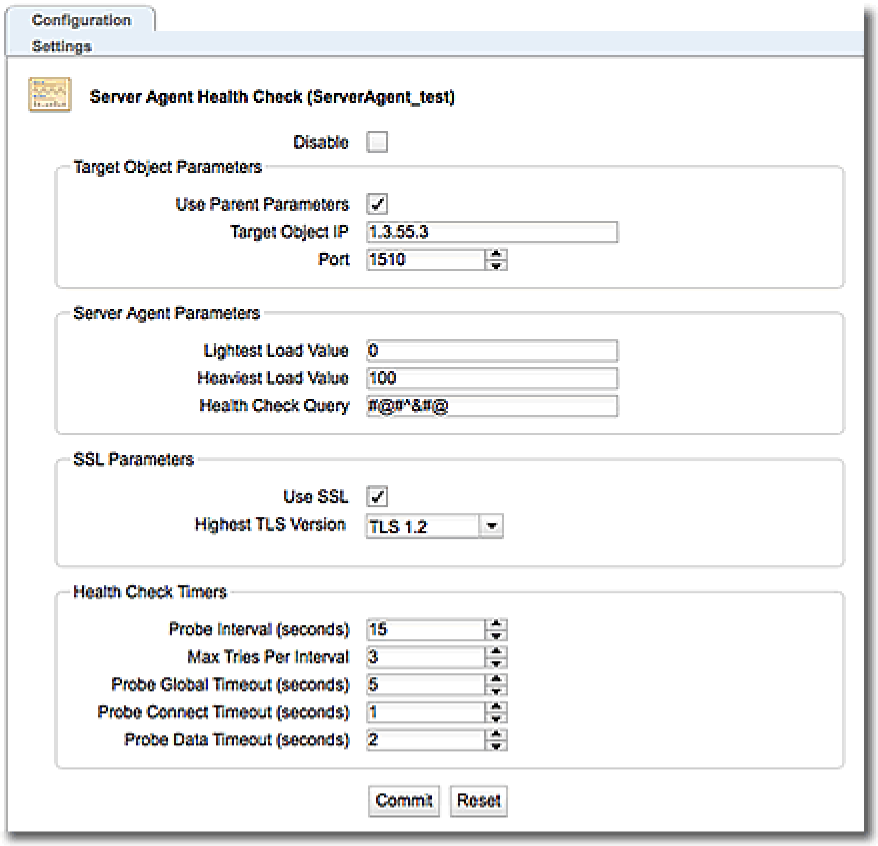
```
eqcli > health_check ServerAgent_test type srvagt ip 1.3.55.3 query
        #@#^&#@ flags probe_ssl

eqcli: 12000287: Operation successful

eqcli > show health_check ServerAgent_test
Health Check Name     : ServerAgent_test
Type                  : srvagt
```

```
Access Parameter      : parent
IP                    : 1.3.55.3
Port                  : 1510
Light Load            : 0
Heavy Load            : 100
Probe Interval        : 15
Max Tries/Interval    : 3
Global Timeout        : 5
Connection Timeout    : 1
Data Timeout          : 2
Highest TLS Version   : 4
Query                 : #@#^&#@
Flags                 : probe_ssl
eqcli >
```

3. Enter additional **`parameter_names`** and **`values`** described in the Server Agent Health Check Parameters table above.

Additional information on using the health check CLI commands can be found in "Health Check Commands" on page 189.

Configuring Server Agent Health Checks Using the GUI



Server Agent Health Check (ServerAgent_test)

Disable	☐

Target Object Parameters

Use Parent Parameters	✓
Target Object IP	1.3.55.3
Port	1510

Server Agent Parameters

Lightest Load Value	0
Heaviest Load Value	100
Health Check Query	#@#^&#@

SSL Parameters

Use SSL	✓
Highest TLS Version	TLS 1.2

Health Check Timers

Probe Interval (seconds)	15
Max Tries Per Interval	3
Probe Global Timeout (seconds)	5
Probe Connect Timeout (seconds)	1
Probe Data Timeout (seconds)	2

5. Modify the Server Agent probe parameter shown in the Server Agent Health Check Parameters table above as necessary.
6. Click on **Commit**.

Refer to "Attaching Health Checks to Load Balancing Objects" on page 592 for instructions on attaching the health check.

Attaching Health Checks to Load Balancing Objects

Health checks can be attached to Servers, Server Pools, Server Instances and LLB Gateways using either the CLI or the GUI.

Each attached health check features an **Optional** option. This is typically used when multiple health checks are attached to an object. For example, if you have an ACV and a VLB health check attached to a server pool, and one of the health checks on a particular server instance probes DOWN, the server instance will be marked DOWN. If the **Optional** flag is enabled on the attached health check, and the other attached health check is NOT failing, the server instance will be marked UP.

Each attached health check also features a **Disable** option. Enabling this option will disable the attached health check only, and NOT the configured health check that may be attached with other load balancing objects.

You must have previously created the health check or use one of the default health checks (See "System Health Checks" on page 554).

Attaching a Health Check using the CLI

Enter the following in the CLI in the global context. Use a configured health check or one of the system health checks:

```
eqcli > object objectname hc healthcheckname
```

where:

objectname - is the name of the object.

healthcheckname - is the name of the configured health check.

For example:

```
eqcli > srvpool srvpool1 hc icmp
Name        Flags
ICMP        disable, optional

eqcli >
```

Additional information on using the health check CLI commands can be found in the "Health Check Commands" on page 189.

Displaying an Attached Health Check using the CLI

You can display the attached health check by using the following format:

```
eqcli > show object objectname hchealthcheckname
```

where:

objectname - is the name of the object.

healthcheckname - is the name of the health check.

For example, the following is an example of a CLI display with the health checks attached to a server instance (**srv1**) in a server pool (**srvpool1**):

```
eqcli > show srvpool srvpool1 si srv1 hc

Name        Flags
Test_TCP
TestICMP    disable, optional

eqcli >
```

Note - Attached ACV and TCP Health Check displays are different. Refer to "Attached ACV and TCP Health Checks and the Test Function" on page 596.

Attaching a Health Check using the GUI

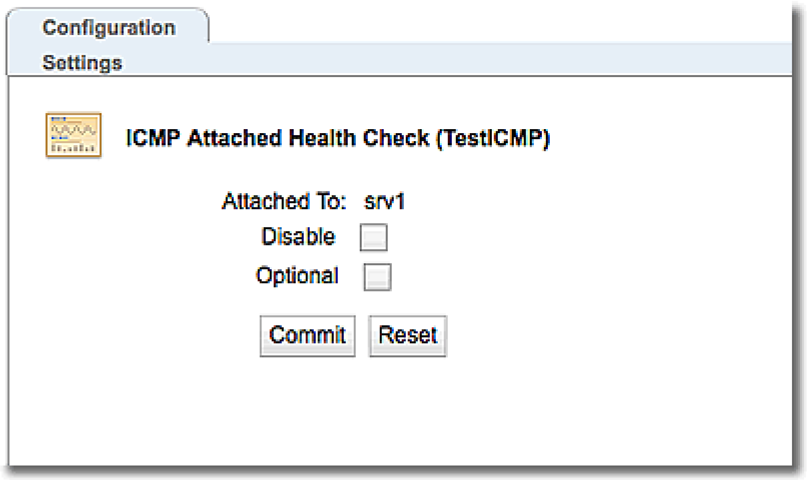
1. Click on the **Load Balance** configuration tab on the left navigational pane on the GUI. If you want to attach an ICMP health check to an LLB Gateway, select the **Link Load Balance** configuration tab, expand the **Outbound** branch and select a configured LLB Gateway.

Perform either step 2 or step 3.

2. Do the following to use the drag and drop functionality:
   a. Select either **Servers**, **Server Pools**, a **Server Instance**, or an **Outbound Gateway** from the left navigational pane. Right click on the **Health Checks** branch of the object and select **Add Health Check.**
   b. From the Health Checks branch on the left navigational pane, select a configured health check and drag and drop it on a **Server, Server Pool**, **Server Instance**, or **Outbound Gateway**.
   c. Click on **Commit** to attach the selected health check(s). It should appear on the object's **Health Checks** branch on the left navigational pane.
3. Do the following to attach a health check using the summary screen:
   a. Click on the object's **Health Checks** branch on the left navigational pane to display the **Health Checks Summary** screen on the right.
   b. Click on + to display a pop up list of configured health checks. The health checks listed are available, yet NOT attached to the object select
   c. Using the check boxes, select one more of the health checks from the **Attach Health Check** screen.
   d. Click on **Commit** to attach the selected health check(s). It should appear on the object's **Health Checks** branch on the left navigational pane and also in the summary screen.

Displaying an Attached Health Check in the GUI

After you attach the health checks to the load balancing objects you can be view the **Attached Health Check** screen on the GUI by selecting **Load Balance > [object] > [object] Health Checks** and then selecting the health check on the left navigational pane. When a health check is selected, its "instance" is displayed on the right configuration pane. An example of an **ICMP Attached Health Check** is displayed as follows:



ICMP Attached Health Check (TestICMP)

Attached To: srv1

Disable ☐

Optional ☐

The attached Health Check screen can be used to:

- **Disable** the health check for the object (not globally)
- Set an **Optional** flag, where if the health check returns a DOWN response, servers will not be marked DOWN if other attached health checks return UP responses.

Click on **Commit** to save your selections or **Reset** to remove any changes.

Note - Attached ACV and TCP Health Check screens are different. Refer to "Attached ACV and TCP Health Checks and the Test Function" on page 596.

Attached ACV and TCP Health Checks and the Test Function

In addition to the **Disable** and **Optional** attached health check options, attached ACV and TCP health checks feature a test feature. The **ACV Test** feature utilizes the query and response strings configured in the health check to check the UP/DOWN status of the selected server, or servers, where server instances are configured in the server pool. The **TCP Test** feature probes the server instance selected and returns UP/DOWN status of the server.

This feature is available on health checks attached to Server Pools or Server Instances.

Using the ACV and TCP Health Check Instance Test Feature in the CLI

When a health check is attached to a server pool, the server name can be specified after `test` in the CLI syntax. This represents a server instance. If a server name is specified, only that server instance on the server pool will be tested. If no server name is specified, all server instances will be tested.

The following is an example of an attached ACV health check:

```
eqcli sp-srv*> show hc ACV
This health check instance is enabled.
Health Check Name  : ACV
Flags              : optional
Server Instance  HC Status
srv1             Up
eqcli sp-srv*>
```

When a health check is attached to a server instance in a server pool, the server name does not need to be entered as the test will be performed on the server instance to which it is attached.

To use the ACV and TCP Test feature in the CLI enter the following.

```
eqcli > sp-srv*-hc-nam*> test servername
```

where:

servername - is the name of the server to which the health check is attached.

If the health check is attached to a server instance, the `test` command would be used in the health check context for the server instance.

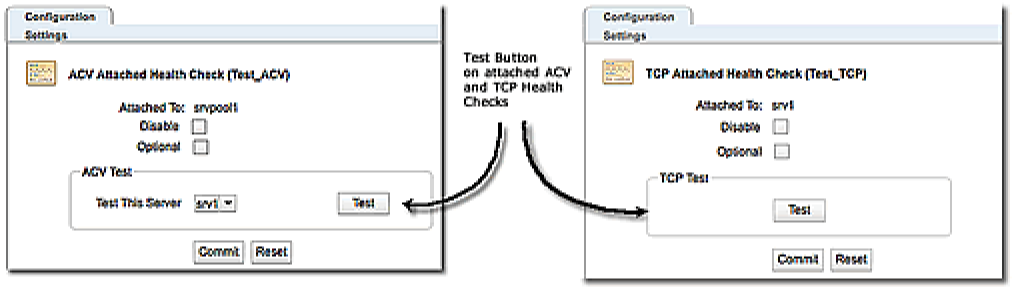
Using the ACV and TCP Health Check Instance Test Feature in the GUI

Select an attached ACV or TCP health check instance on the left navigational pane on the GUI. The health check instance **Configuration Settings** screen will be displayed on the right configuration pane.

If an ACV or TCP health check is attached to a *server pool*, use the **Test This Server** drop down list to select a server instance or all of the server instances on the server pool.. Click on the **Test** button to begin the test.

If an ACV health check is attached to a *server instance*, the **Test** button only will be available as the test will be performed on the server instance to which it is attached.

A results pop up will be displayed upon completion of the test that displays the status of the selected server(s).



Click on **Commit** to save your selections or **Reset** to remove any changes.

Displaying Configured Health Checks

All of your configured health checks can be displayed using either the CLI or the GUI.

Viewing All Configured Health Checks using the CLI

To view the status of all of the configured health checks using the CLI enter the following at the `eqcli >` prompt:e

```
eqcli > show health_check
Name          Type    IP  Status
ICMP-Default  icmp        --
TCP-Default   tcp         --
hc_srvagt     srvagt      --
hc_acv        acv         In Use
hc_vlb        vlb         Disabled, Attached
hc_udp        udp         In Use
eqcli >
```

where:

`Name` - is the name that you assigned to the health check

`Type` - is the type of health check (i.e., `icmp`, `vlb`,`acv`)

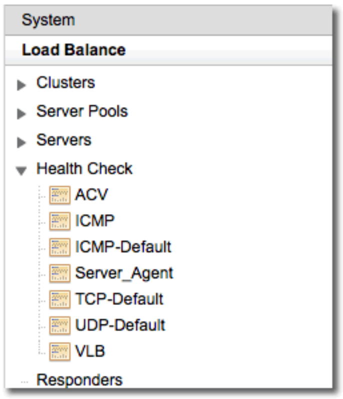
`IP`- is the IP address of the object being queried, if it is specified in the health check configuration..

`Status` - is the status of the health check. Options are:

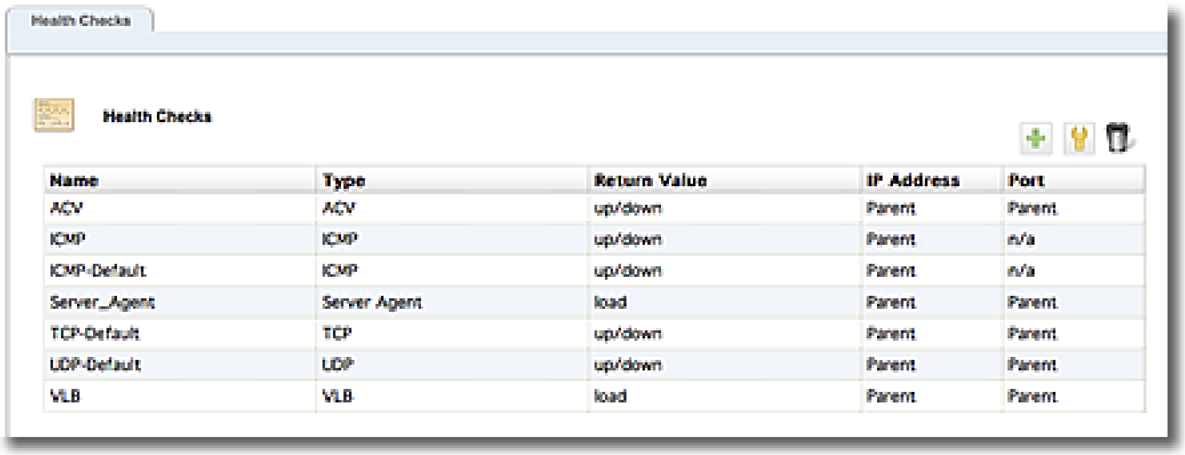
> **`incomplete`** - indicates that the health check has been created, however it is missing parameters such as `ip` or `port`.
>
> **`disabled, attached`** - indicates that the health check has been globally disabled, however, a health check instance is attached to a load balancing object.
>
> **`disabled`** -indicates that the health check is globally disabled. All attached health check instances of this health check will also be disabled.
>
> **`In use`** - indicates that the health check is configured and one or more instances of the health check are attached to a load balancing object.

To view all of the configured health checks using the GUI:

1. Log in to the GUI.
2. Select the **Load Balance** configuration tab on the left navigational pane if it is not already selected.
3. Click on **Health Checks** on the left navigational pane to expand the branch. This displays all of the configured health checks. For example::



4. View the status of all of the configured health checks by selecting **Health Check.** The following will be displayed on the right configuration pane.



Health Checks

Name	Type	Return Value	IP Address	Port
ACV	ACV	up/down	Parent	Parent
ICMP	ICMP	up/down	Parent	n/a
ICMP-Default	ICMP	up/down	Parent	n/a
Server_Agent	Server Agent	load	Parent	Parent
TCP-Default	TCP	up/down	Parent	Parent
UDP-Default	UDP	up/down	Parent	Parent
VLB	VLB	load	Parent	Parent

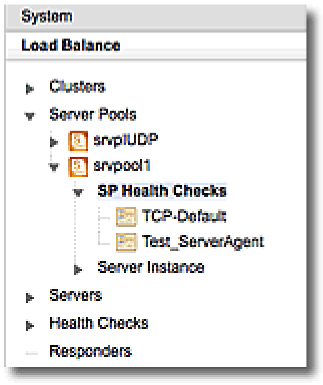
Double clicking on any of the configured health checks will display the health check configuration screens for each health check, where you can modify it. You can also select the health check and then click on the icon.

You can delete a health check by selecting the health check and clicking on the icon, or dragging and dropping the health check to the icon.

You can add a new health check from this screen by selecting the **+**.

Viewing All Attached Health Checks using the GUI

Click on any of the Load Balance objects on the tree to expand the branch. All of the health checks that are attached to the object will be displayed in the object's **Health Checks** branch. For example:



Displaying the Status of Attached Health Checks using the GUI

Select the object's **Health Checks** branch in the left navigational pane. The **Health Check Summary** screen will appear on the right configuration pane.

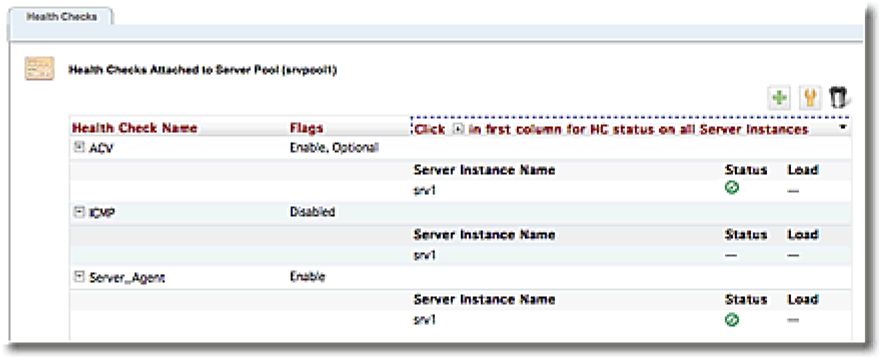
The following are examples of health check summary displays for health checks attached to server pools, server instances, servers, and LLB gateways.

For each of the health check displays described below:

- The ✓ icon in the **Status** column indicates that the health check probe is UP.
- The ● icon in the **Status** column indicates that the health check probe is DOWN.
- Double click on any of the attached health checks to display the attached health check screen where you can set the optional and disable flags. You can also select the health check and then click on the icon.
- Delete an attached health check by selecting the attached health check and clicking on the icon, or dragging and dropping the health check to the icon.
- Attach a new health check from this screen by selecting the + icon. (Refer to"Attaching Health Checks to Load Balancing Objects" on page 592 for more information).

Displaying the Status of Attached Server Pool Health Checks

Select **Load Balance > Server Pool > Health Checks** on the left navigational pane. The following is an example of the display on the right configuration pane. The + icon in the first column expands to present the status/load of each health check on each server instance.



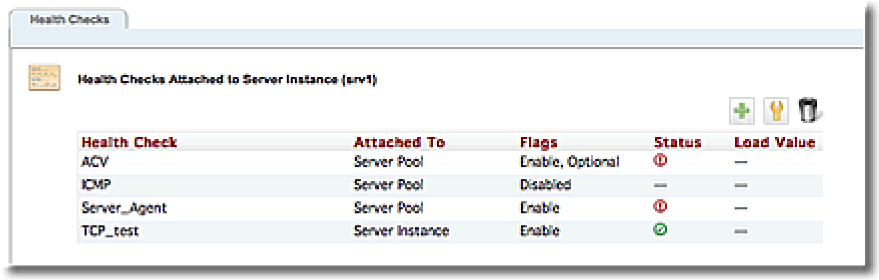
Health Checks

Health Checks Attached to Server Pool (srvpool1)

Health Check Name	Flags	Click ⊞ in first column for HC status on all Server Instances		
⊟ ACV	Enable, Optional			
		Server Instance Name	Status	Load
		srv1	✓	—
⊟ ICMP	Disabled			
		Server Instance Name	Status	Load
		srv1	—	—
⊟ Server_Agent	Enable			
		Server Instance Name	Status	Load
		srv1	✓	—

Displaying the Status of Attached Server Instance Health Checks

Select **Load Balance > Server Pools> Server Instance > [Select a Server Instance] > SI Health Checks** on the left navigational pane. The following is an example of the display on the right configuration pane:



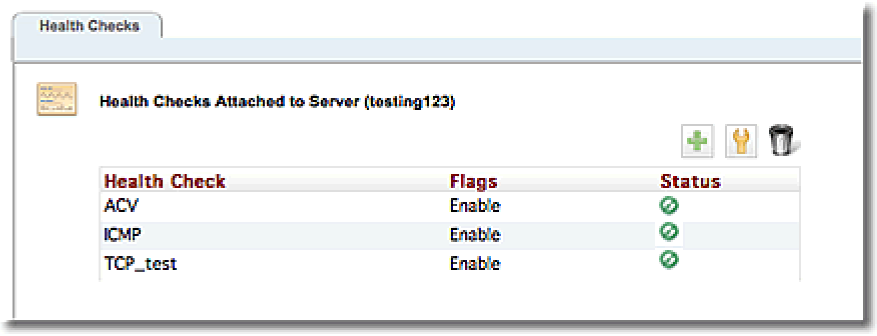
Health Checks

Health Checks Attached to Server Instance (srv1)

Health Check	Attached To	Flags	Status	Load Value
ACV	Server Pool	Enable, Optional	ⓘ	—
ICMP	Server Pool	Disabled	—	—
Server_Agent	Server Pool	Enable	ⓘ	—
TCP_test	Server Instance	Enable	✓	—

Displaying the Status of Attached Server Health Checks

Select **Load Balance > Servers > [Select a Server] > Server Health Checks** on the left navigational pane. The following is an example of the display on the right configuration pane:



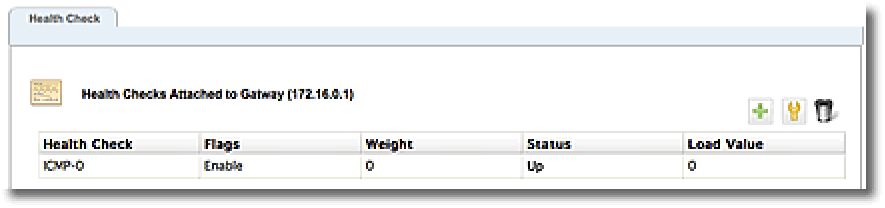
Health Checks

Health Checks Attached to Server (testing123)

Health Check	Flags	Status
ACV	Enable	✓
ICMP	Enable	✓
TCP_test	Enable	✓

Displaying the Status of Attached LLB Gateway Health Checks

Select Link Load Balance > Outbound > Gateways . > [Select a Gateway] > Gateway Health Checks on the left navigational pane. The following is an example of the display on the right configuration pane:



Health Checks Attached to Gatway (172.16.0.1)

Health Check	Flags	Weight	Status	Load Value
ICMP-0	Enable	0	Up	0

Disabling Health Checks

By default, all health checks are enabled when a health check instance is added to a server, server pool, server instance, or LLB Gateway.

Health checks can be disabled by either:

- Disabling a health check attached to a server, server pool, server instance, or LLB Gateway or
- Disabling globally by accessing the health check object and selecting the "disable" option.

You can use the CLI or the GUI to disable health checks.

Disabling Using the CLI

To disable the health check on a *global* level using the following syntax:

```
eqcli > health_check healthcheckname flags disable
```

where:

healthcheckname - is the name of the health check to be globally disabled.

For example:

```
eqcli > health_check ICMP flags disable

eqcli > show health_check ICMP
Health Check Name     : ICMP
Type                  : icmp
Access Parameter      : parent
IP                    :
Probe Interval        : 15
Max Tries/Interval    : 3
Flags                 : disable
eqcli >
```

To disable an *attached* health check enter the following:

```
eqcli > object objectname hc healthcheckname flags disable
```

where:

objectname - is the name of the load balancing object

healthcheckname - is the name of the health check to be globally disabled.

For example:

```
eqcli > show srvpool srvpool1 hc ICMP flags disable
eqcli: 12000287: Operation successful

eqcli > show srvpool srvpool1 hc ICMP

Attached Health Check is disabled
Health Check Name  : ICMP
Flags              : disable

Server Instance  HC Status
srv1             N/A
eqcli >
```

Disabling Using the GUI

Disable the health check *globally* in the GUI as follows:

1. Log in to the GUI.
2. Click on the **Load Balance > Health Checks** on the left navigational pane.
3. Select a health check to display the **Settings** screen on the right configuration pane.
4. Select the **Disable** check box.
5. Click on the **Commit** button to save the change.

Disable *attached* health checks in the GUI as follows:

1. Log in to the GUI.
2. If a health check instance is attached to a server, server pool, or server instance, click on the **Load Balance** configuration tab on the left navigational pane. If a health check instance is attached to an LLB Gateway, click on the **Link Load Balance** tab.
3. On the left navigational pane, expand the branch for the object to display the health check instance icon.
4. Click on the health check instance to display the **Settings** display on the right configuration display.
5. Click on the **Disable** check box.
6. Click on the **Commit** button to save the change.

Deleting Health Checks

Health checks can be deleted using the CLI or the GUI. They can be deleted by either:

- Deleting a health check attached to a server, server pool, server instance, or LLB Gateway or
- Deleting the health check object

The ICMP-Default, TCP-Default, or UDP-Default System health checks cannot be deleted.

Delete Using the CLI

To delete an attached health check using the CLI, use the following format:

```
eqcli > object objectname no health_check healthcheckname
```

where:

objectname- is the name of the load balancing object to which the health check is attached.

healthcheckname - is the name of the attached health check.

For example

```
eqcli > srvpool srvpool1 no health_check TCP_Test
eqcli: 12000287: Operation successful
eqcli >
```

To delete a health check object using the CLI,

```
eqcli > no health_check healthcheckname
```

where:

healthcheckname - is the name of the health check being deleted.

For example:

```
eqcli > no health_check TCP_Test
eqcli: 12000287: Operation successful
eqcli >
```

Delete Using the GUI

Delete a health check using the GUI from *either* the left navigational pane or the right configuration pane:

To delete an attached health check:

From the left navigational pane:

1. Select **Load Balance > Health Checks**.
2. Select a health check that you would like to delete.
3. Right click and select **Delete Health Check**.

From the Health Checks list on the right configuration pane:

1. Select **Load Balance > Health Checks**.on the left navigational pane.
2. Select a health check from the list on the right configuration pane.
3. Click on the icon.

To delete a health check object:

1. Select the load balancing object on the left navigational pane.

From the left navigational pane:

2. Select an attached health check that you would like to delete.
3. Right click and select **Delete Health Check**.

From the Health Checks list on the right configuration pane:

1. Select an attached health check on the left navigational pane.
2. From the list on the right configuration pane, select the attached health check.
3. Click on the icon.

Probe Coalescing

If two or more attached health checks of the same type have the same probe parameters described below, they will coalesce into a single probe.

ICMP Health Checks

All ICMP probing occurs at the same frequency, which is the highest frequency needed to satisfy the highest probe interval/tries combination of all ICMP probes across all objects. If the **Target IP Addresses** (`ip`) of two attached ICMP health checks are the same, they will be combined into a single probe.

TCP Health Checks

If the **Target Object IP** (`ip`) **Port** (`port`),**Probe Global Timeout** (`probe_gto`), **Probe Connect Timeout** (`probe_cto`), **Probe Data Timeout** (`probe_dto`) parameters, and **Use SSL** (`probe_ssl`) option of two attached TCP health checks are the same, they will be combined into a single probe. If the **Probe Interval** (`probe_interval`) and **Max Tries per Interval** (`probe_maxtries`) parameters are different, probing will occur at the highest frequency of all health checks that are being serviced by the probe.

UDP Health Checks

If the **Target Object IP** (`ip`) **Port** (`port`), and **Probe Global Timeout** (`probe_gto`)parameters of two attached health checks are the same, they will be combined into a single probe. If the **Probe Interval** (`probe_interval`) and **Max Tries per Interval** (`probe_maxtries`) parameters are different, probing will occur at the highest frequency of all health checks that are being serviced by the probe.

ACV Health Checks

If the **Target Object IP** (`ip`) **Port** (`port`),**Probe Global Timeout** (`probe_gto`), **Probe Connect Timeout** (`probe_cto`), **Probe Data Timeout** (`probe_dto`) parameters are the same, the **Use SSL** (`probe_ssl`) option of two attached ACV health checks are the same, and the **ACV Query** (`query`) and **ACV Response** (`response`) parameters of the two attached ACV health checks are the same.they will be combined into a single probe. If the **Probe Interval** (`probe_interval`) and **Max Tries per Interval** (`probe_maxtries`) parameters are different, probing will occur at the highest frequency of all health checks that are being serviced by the probe. .

VLB Health Checks

If the **VLB Manager** (`vlb_mgr`) and **UUID** (`vlb_uuid`) parameters on two attached VLB health checks are the same, they will be combined into a single probe. A single probe can access both the VMs CPU and RAM information, and appropriately manage either CPU or RAM (or both) VLB health check instances . If the **Probe Interval** (`probe_interval`) and **Max Tries per Interval** (`probe_maxtries`) parameters are different, probing will occur at the highest frequency of all health checks that are being serviced by the probe.

Server Agent Health Checks

If the **Target Object IP** (`ip`) **Port** (`port`),**Probe Global Timeout** (`probe_gto`), **Probe Connect Timeout** (`probe_cto`), **Probe Data Timeout** (`probe_dto`) parameters are the same, and the **Use SSL** (`probe_ssl`) option and **Health Check Query** (`query`) parameters of the two attached Server Agent health checks are the same for two attached Server Agent health checks, they will be combined into a single probe. If the **Probe Interval** (`probe_interval`) and **Max Tries per Interval** (`probe_maxtries`) parameters are different, probing will occur at the highest frequency of all health checks that are being serviced by the probe.

When a single probe probes on behalf of more than one attached health check, an individual attached health check is marked DOWN according to the interval value associated with it. If the time stamp on the last good probe has not occurred within the last interval seconds, then the probe will be marked DOWN. For example, if two TCP health checks attached to two server instances have the same **Target Object IP** (`ip`) **Port** (`port`),**Probe Global Timeout** (`probe_gto`), **Probe Connect Timeout** (`probe_cto`), **Probe Data Timeout** (`probe_dto`) parameters and the **Max Tries per Interval** (`probe_maxtries`) parameter is 3, yet one has probe **Probe Interval** (`probe_interval`) of 60 seconds and the other 15 seconds. They will be combined into a single probe that sends a probe every 5 seconds (Prove interval=15/Max tries per interval =3). If the probe fails, 15 seconds later one of the two attached health checks will be marked DOWN (because its interval is 15), and 60 seconds after the probe fails, the 2nd attached health check will be marked down (because its interval is 60).

Health Check Timeouts

Health check timeouts are configured as part of each health check's configuration. The types of timeouts vary, based on the health check protocol, however, their basic functionality is the same.

Timeouts were discussed previously in the health check configuration parameters. The purpose of this section is to provide further descriptions for better understanding of how the timeouts and intervals work with one another.

ICMP Health Check Timeouts

By default, Equalizer sends an Internet Control Message Protocol (ICMP) echo request (commonly called a “ping”) to the IP address of every configured server object. The timeouts that control ICMP health check probes are determined by the health check definition.

ICMP Health Check Parameters (CLI Parameter)	**Minimum**	**Default**	**Maximum**	**Units**
Max Tries Per Interval(**probe_maxtries**)	> 1	3	30	integer
Probe Interval (**probe_interval**)	0	15.0	60.0	seconds

By default, ICMP health check probes are sent a maximum of 3 times every 15 seconds to every configured server that has Layer 3 probes enabled. Within a 15-second probe interval, the delay between successive ping requests to the same server is determined internally -- it can be as short as one second to a server that is not responding to ICMP requests and can be 5 seconds or longer when a server is responding to ICMP echo requests, depending on the amount of traffic that Equalizer is currently processing.

If a server responds to an ICMP Health Check, it is marked "Layer 3 UP". If a server does not respond to an ICMP echo request, it will be marked "Layer 3 DOWN" by Equalizer only if the server has responded to at least one ICMP echo request since Equalizer was last rebooted. This behavior accounts for the fact that many servers are configured by default to never respond to ICMP echo requests as a security precaution. In other words, if a server has never responded to a Layer 3 health check probe since the last reboot, it is never marked "Layer 3 Down".

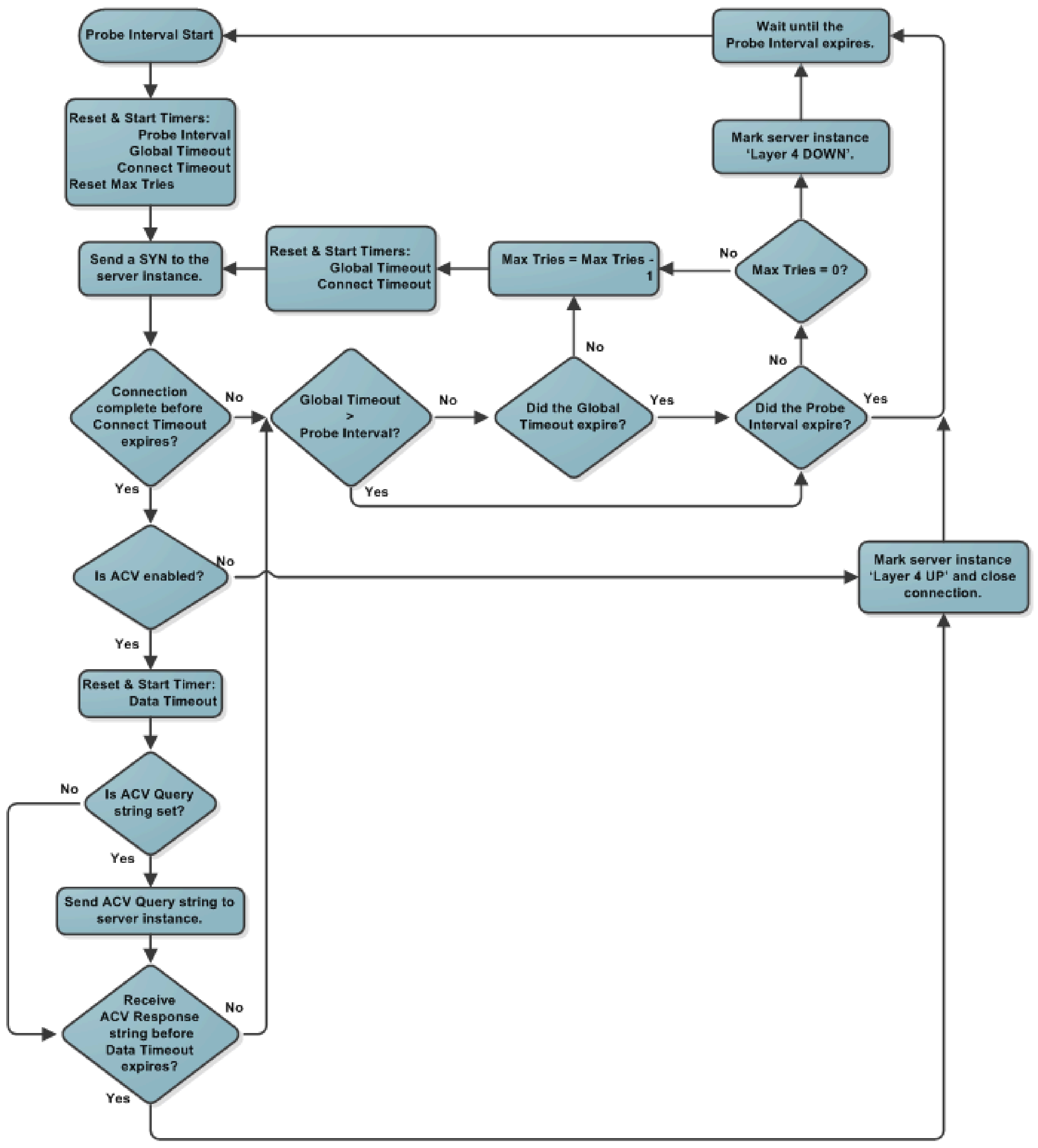
Note - Responding to ICMP echo requests is an option on most server platforms. If ICMP echo reply is disabled on one or more of the servers in your configuration, then you may want to disable ICMP echo requests on Equalizer to reduce traffic between Equalizer and the servers, and rely solely on the other probing mechanisms.

Layer 4 TCP and ACV Health Check Timeouts

By default, Equalizer sends TCP Health Checks to every server instance in a TCP server pool. Equalizer and the server exchange a three-way TCP handshake to open a TCP connection. If ACV is also configured, the ACV probe takes place after the TCP connection is established. Once the handshake is complete and ACV data is optionally exchanged, the probe connection is closed.Once enabled, TCP and ACV probe behavior is determined by the timeouts determined by the health check definition.

GUI Parameter (CLI Parameter)	Minimum	Default	Maximum	Units
Max Tries Per Interval (**max_tries**)	>1	3	30	integer
Probe Interval (**probe_interval**)	1	15	60	seconds
Probe Global Timeout (**probe_gto**)	0	5	120	seconds
Probe Connect Timeout (**probe_cto**)	0	1	60	seconds
Probe Data Timeout (**probe_dto**)	0	2	60	seconds

By default, Equalizer sends Layer 4 health check probes to a server instance at most 3 times (the Max Tries Per Interval setting), every 15 seconds (the Probe Interval). Exactly how many probes are sent in any given probe interval is determined by how long it takes each probe to complete and whether any of the timeouts listed above expire while Layer 4 health checks are being sent and received. Both TCP and ACV probes are subject to the same set of timeout values, as summarized in the following flowchart.


Probe Interval Start
Reset & Start Timers: Probe Interval Global Timeout Connect Timeout Reset Max Tries
Send a SYN to the server instance.
Reset & Start Timers: Global Timeout Connect Timeout
Max Tries = Max Tries - 1
Max Tries = 0?
Mark server instance 'Layer 4 DOWN'.
Wait until the Probe Interval expires.
Connection complete before Connect Timeout expires?
Global Timeout > Probe Interval?
Did the Global Timeout expire?
Did the Probe Interval expire?
Is ACV enabled?
Mark server instance 'Layer 4 UP' and close connection.
Reset & Start Timer: Data Timeout
Is ACV Query string set?
Send ACV Query string to server instance.
Receive ACV Response string before Data Timeout expires?
Yes
No

Server Agent and VLB Health Check Timeouts

Server Agent and VLB health checks behave the same as the timeouts for Layer 4 TCP and ACV health checks in the previous sections, with the exception that the Probe Data Timeout (probe_dto) is the timeout for the server response for these health checks rather than ACV. This affects only the part of the flowchart that is outlined in the previous sections.

VLB health checks use the VLB Manager timeout value.

Chapter 17

Automatic Cluster Responders

Note - Responders are not supported on E250GX model Equalizers.

Sections within this chapter include:

Automatic Cluster Resopnder Overview

Note -Responders are not supported on E250GX model Equalizers.

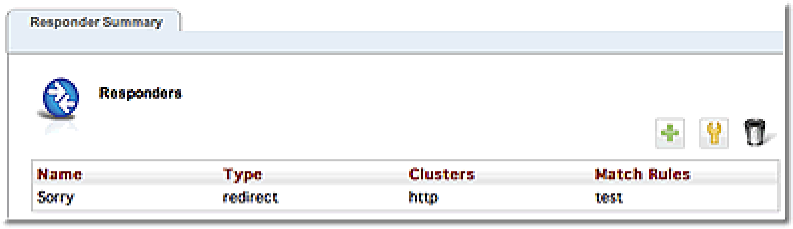
A Responder is a server-like object that can be associated with a Match Rule. It provides you with the ability to cleanly load balance traffic where server pools associated with a cluster are not available to satisfy a client's request. The feature extends Cluster Match Rules to allow them to specify a target Responder which is used to provide a response to the client when the server pool in the match rule is not available. If an incoming request matches a Match Rule expression and the server pool specified in the Match Rule is down, a Responder definition in the Match Rule (if present) tells Equalizer to send one of two automatic responses to the client:

- **A customized HTML “sorry page”** that can, for example, ask the client to retry later or go to another URL.
- **A standard HTTP Redirect response** that specifies a return code and redirect URL. When the client receives this page, it is automatically redirected to the redirect URL. Redirect pages can be configured to use parts of the request URL in the HTTP Redirect response (using a regular expression).

Responder Summary

The **Responder Summary** screen displays the names of configured responders, **Type**,associated **Clusters** and **Match Rules**.

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Load Balance** configuration tab if it is not already selected.
3. Click on **Responders**to display the **Responder Summary** screen.



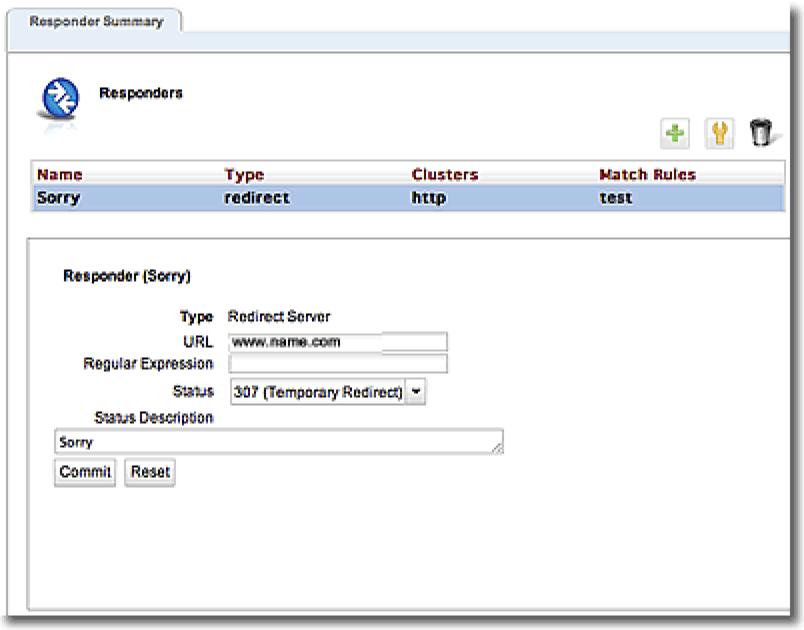
Name	Type	Clusters	Match Rules
Sorry	redirect	http	test

From this screen you can add a new responder by clicking on "+" .

You can delete a server by selecting a responder and clicking on .

You can modify server configuration by selecting a responder and clicking on . This will

activate expand the Responders summary screen to include Responder Configuration details described in "Adding a Responder" on page 616.



Name	Type	Clusters	Match Rules
Sorry	redirect	http	test

Responder (Sorry)

Type	Redirect Server
URL	www.name.com
Regular Expression	
Status	307 (Temporary Redirect)
Status Description	Sorry

Managing Responders

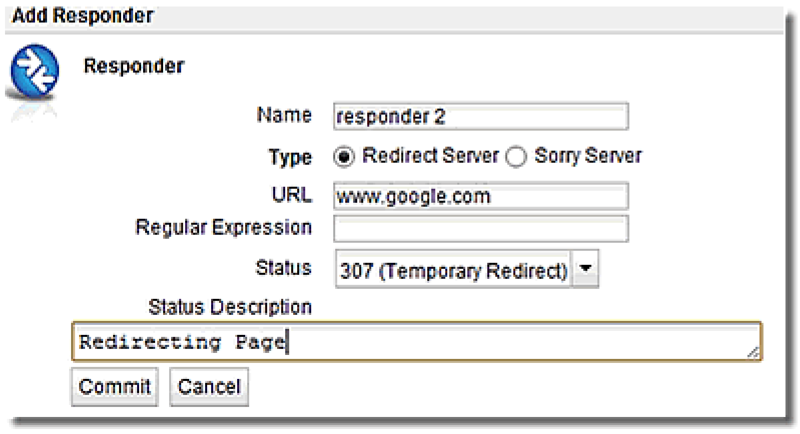
To display a list of all currently defined Responders, select the Load Balance configuration tab on the left navigational pane and click on the arrow (▶) beside **Responders**to expand the branch. Select any of the **Responders** on the tree to display their configuration on the right pane.

- To add a Responder, right click on Responders on the left navigational pane and select **Add Responder.**
- To edit a Responder's configuration, click on the specific **Responder** on the **Responder** branch on the navigational pane to display the configuration on the right.
- To delete a Responder, click on the specific **Responder** on the **Responder** branch on the left navigational pane and select **Delete Responder** A Responder cannot be deleted if it is currently used in a match rule definition.

Adding a Responder

Responders are a "global" resource: once created, they can be individually assigned to one or more match rules in one or more clusters. Up to 8192 Responders can be created.

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Load Balance** configuration tab if it is not already selected.
3. Right click on **Responder** on the left navigational pane and select **Add Responder.** The **Add New Responder** dialog appears. By default, the form for creating a **Redirect Responder** is displayed:


Add Responder

Responder

Name	responder 2
Type	[✓ Redirect Server] Sorry Server
URL	www.google.com
Regular Expression	
Status	307 (Temporary Redirect)
Status Description	Redirecting Page

2. Type a **Name** for the Responder or leave the default name provided.
3. Do one of the following:
   - Create a custom HTML page by selecting **Sorry Server.** The dialog changes to a text entry box, into which you can type the HTML that Equalizer will return to clients. The text size limit is 4096 bytes.
   - Create a standard **Redirect** page by supplying the following information in the pop up screen:

Status	The HTTP status code to return to the client. The default return code is **307 (Temporary Redirect).** Use the drop-down box to choose a different return code: 301 (Moved Permanently) 302 (Found) 303 (See Other)

URL	The HTTP Redirect URL: the full URL of the page to which the client will be redirected, as in the following example:
	`http://www.sitename.com/redirect/redirect.html`
	If a **Regular Expression** is used to split the client URL into string variables, any variables appearing in the URL are replaced with strings from the request URL. The following is an example of a Redirect URL with named variables:
	`http://$1.$2.net$3$4`
	
	The maximum size of a redirect URL is 1200 characters
	
	See the see "Using Regular Expressions in Redirect Responders" on page 618.

Regular Expression	An optional POSIX-style regular expression that splits the incoming request URL into variables that can be used for string replacement in the HTTP Redirect **URL** (see above). See "Using Regular Expressions in Redirect Responders" on page 618.

When you are done, click on **Commit**.

4. In the screen that follows, you can optionally test your responder. Do one of the following:
   - For a **Sorry Server** responder, click the **test** button to see a preview of the page. Click the **close** button to close the preview.
   - For a **Redirect** responder, enter a **Test** URL (or use the default) and click the **test** button to see how the regular expression breaks the test URL into variables for re-use in the URL you supplied in the previous step.
   - Click the Next icon (**>**) at the top of the dialog to skip testing.
5. On the next screen, do one of the following:
   - Click the Back icon (**>**) at the top of the screen to review the responder configuration.
   - For a **Sorry Server**, click **commit** to add this responder or **cancel** to close the dialog without adding the responder.
   - For a **Redirect** responder, this screen displays the responder **Redirect URL** and the **Regular Expression** (if supplied).

     If you clicked the **test** button on the previous screen, the **Match Components** and **Resulting Redirect** produced by matching the **Test** URL against the **Regular Expression** are also displayed (any variables appearing in the **Redirect URL** are replaced with strings from the **Test** URL).

6. Click **Commit** to add the responder or **Cancel** to close the dialog without adding the responder.

Modifying a Responder

1. To modify the configuration of an existing Responder, click on a Responder on the **Responder** branch of the left navigational pane in the left frame.The Responder's **Configuration** tab appears.
2. Update the Responder configuration as desired; see "Adding a Responder" on page 616 for a description of all Responder parameters.
3. Click **Commit** to save your changes.

Using Regular Expressions in Redirect Responders

In some cases, it may be desirable to examine the URL of an incoming request and re-use parts of it in the URL returned to the client by a Redirect Responder. This is the purpose of the **Regex** field: specify a custom regular expression that is used to:

- parse the URL of an incoming request
- break it down into separate strings (based on the positions of literal characters in the expression)
- assign each string to a named variable

These named variables can then be used in the URL field of the Redirect Responder. When the Responder replies to a client, it performs string substitution on the URL.

Because the purpose of using regular expressions to perform string substitution in Redirect URLs is to parse request URLs into strings, constructing an appropriate regular expression requires an exact knowledge of the format of the request URLs that will typically be coming in to the cluster IP.

Equalizer supports POSIX-style extended regular expressions.

See the examples that follow below to help you understand how regular expressions are constructed and interpreted by Responders.

Example 1 - HTTPS Redirect

The simplest form of HTTPS redirect involves simply referring the user to the top level of the https:// site, regardless of the path information that may have been included in the original request URL. For example, we could direct all requests for:

```
http://www.example.com/<path>
```

to:

```
https://www.example.com
```

But, this forces the client to re-specify the <path> after the redirect. It would be better to redirect to a URL that includes the path information:

```
https://www.example.com/<path>
```

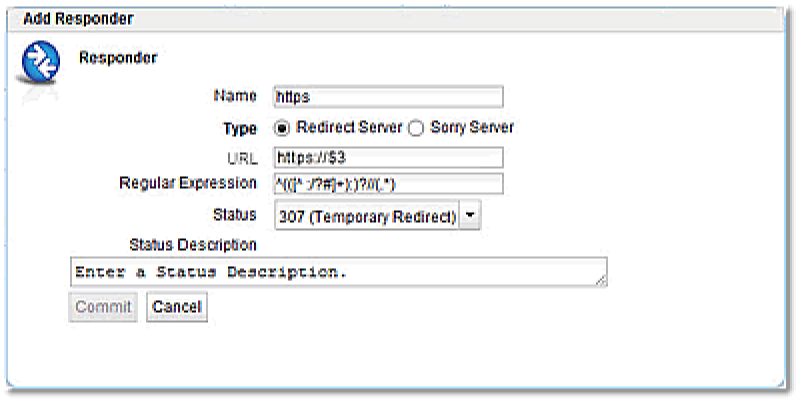
The following regular expression:

```
^(([^ :/?#]+):)?//(.*)
```

breaks a request URL into the following named variables:

```
$0 http://www.example.com/<path>
$1 http
$2 http:
$3 www.example.com/<path>
```

We can then use these variables in the URL field as shown in the following Responder **Configuration** screen (tab):



Name: https
Type: [✓ Redirect Server] Sorry Server
URL: https://$3
Regular Expression: ^(([^ :/?#]+):)?//(.*)
Status: 307 (Temporary Redirect)
Status Description: Enter a Status Description.

This Responder can be used in any cluster where a Redirect to an HTTPS cluster is desired.

Example 2 - Multi-Hostname Redirect

Let’s assume that we have a set of ".com" host names, all of which resolve to the same cluster IP, and we need a Responder that redirects requests to the same hostname prefixes with a ".net" suffix. We also want to include the rest of the URL exactly as specified by the client. For example, we want requests to URLs in these formats:

```
http://www.example.com/<path>
http://www.example2.com/<path>
http://www.example3.com/<path>
```

to be redirected to the following URLs:

```
http://www.example.net/<path>
http://www.example2.net/<path>
http://www.example3.net/<path>
```

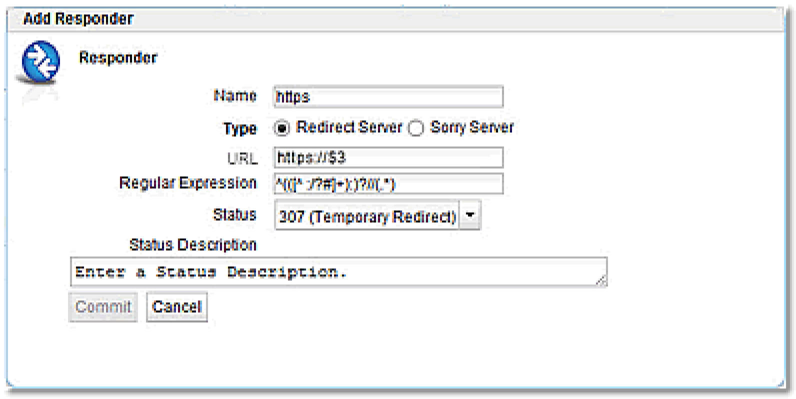
The following regular expression:

```
^(([^ :/?#]+):)?//([^ \r/?#.]+)?\.([^ \r/?#.]+)?\.([^ \r/?#]+)?(/[^ \r]+)?
```

breaks the request URL into the following named variables:

```
$0 http://www.example.com/<path>
$1 http:
$2 http
$3 www
$4 example
$5 com
$6 /<path>
```

We can then use these variables in the URL field as shown in the following Responder **Configuration** screen (tab):



Add Responder

Responder

Name: https

Type: [✓ Redirect Server] Sorry Server

URL: https://$3

Regular Expression: ^(([^ :/?#]+):)?//(.*)

Status: 307 (Temporary Redirect)

Status Description

Enter a Status Description.

It should be noted that this example will not work for requests with destination URLs specified with an IP address for a hostname (e.g.,"12.34.56.78" instead of "www.example.com"). Providing support for IP addresses in URLs as well as DNS host names would involve either: a more complex regular expression that matches both; or, an additional Responder with a regular expression that matches IP addresses, as well as two match rules to match the two types of host names (so that the appropriate Responder replies to the client).

Example 3 - Directory Redirect

The next example involves redirecting requests that include a particular directory to a different domain, omitting the directory from the redirect URL's path. Let's say we want all requests for:

```
http://www.example.com/images/<path>
```

to be redirected to:

```
http://images.example.com/<path>
```

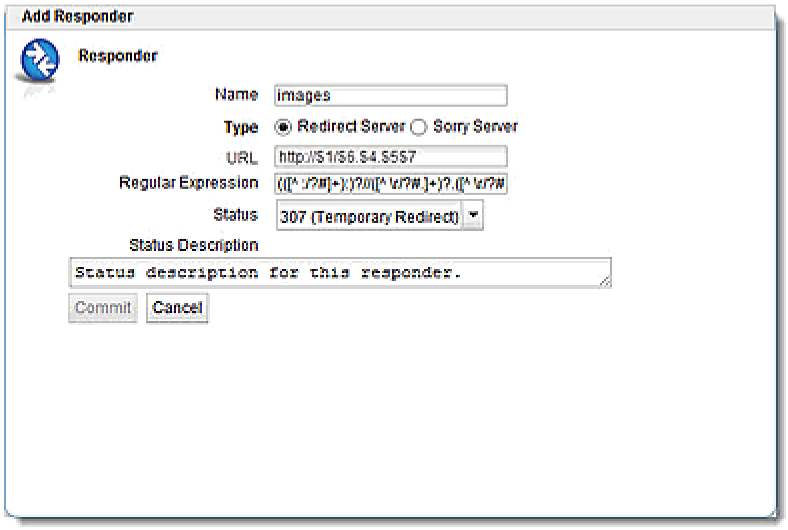
The following regular expression:

```
(([^ :/?#]+):)?//([^ \r/?#.]+)?.([^ \r/?#.]+)?.([^ \r/?#]+)?(/[^ \r]+)?(/[^ \r]+)
```

breaks the request URL into the following named variables:

```
$0 http://www.example.com/images/<path>
$1 http
$2 http:
$3 www
$4 example
$5 com
$6 /images
$7 /<path>
```

We can then use these variables in the URL field as shown in the following Responder **Configuration** screen (tab):


Add Responder

Responder

Name: images

Type: [✓ Redirect Server] Sorry Server

URL: http://$1/$6.$4.$5$7

Regular Expression: (([^ :/?#]+):)?//([^ \r/?#.]+)?.([^ \r/?#

Status: 307 (Temporary Redirect)

Status Description:

`Status description for this responder.`

This Responder can be used in a Match Rule in any cluster where a similar directory name based redirect is required.

Using Responders in Match Rules

Once a responder is created, it can be associated with a cluster using a match rule (See "Using Match Rules" on page 428). When adding a responder to a match Rule, the way the match rule is configured has a direct effect on the conditions under which the responder is used:

expression:	The default match rule **expression** [any()] matches all incoming requests. If you want the responder to be used only for specific requests, then create an appropriate match rule expression to match those requests; See "Using Match Rules" on page 428.
server selection:	By default, no servers are selected in a match rule. This means that any incoming request URL that matches the match rule expression will be handled by the responder specified in the match rule. If you want the responder to be used only if no servers (or particular servers) are available, select all (or some) of the **servers** listed in the match rule **Configuration** screen (tab).

Once a responder is created, it can then be selected in a match rule’s **response** list. The following sections show some common match rule and Responder configurations.

Creating a Match Rule for a “Sorry Page”

The most common use of a responder is to change the default match rule behavior when no server pools are available in a cluster. By default, every HTTP and HTTPS cluster is created with a **Default** match rule that does not specify a Responder -- thus, if all the server pools in the **Default** match rule are down, Equalizer drops the client connection to the cluster.

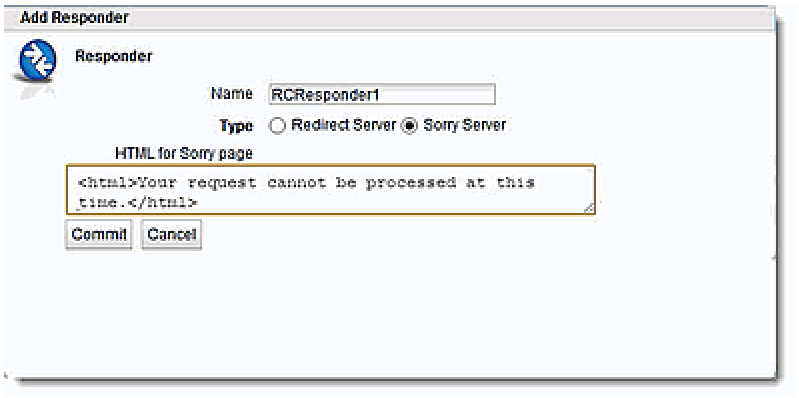
In order to change the default behavior and supply a “sorry page” or redirect for a cluster, you need to add a new match rule that:

- matches any incoming request
- selects the server pool specified

- has a **Sorry Server** Responder selected

For example, let's say you have two Responders defined and there is an existing cluster that you would like to redirect to http://www.example.com when no server pools in the cluster are available. To accomplish this, we need to create a new Responder and then add a match rule to the cluster:

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Load Balance** configuration tab if it is not already selected.
3. Right click on **Responder** on the left navigational pane and select **AddResponder.** The **Add New Responder** dialog appears.



Name: RCResponder1

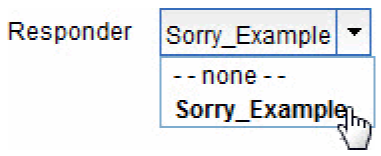
Type: ○ Redirect Server ● Sorry Server

HTML for Sorry page: `<html>Your request cannot be processed at this time.</html>`

4. Type `Sorry_Example` into the **Name** field and select **Sorry Server**.
5. Type the HTML content for the page to display into the text box that appears, as shown in the following example.:
6. Click **Commit** to save the new Responder.
7. Right-click on the name of the cluster for which you want to display the sorry page in the left frame and select **Add Match Rule.**
8. Refer to "Creating a New Match Rule" on page 445 to configure the match rule. Leave the match rule **expression** set to the default [any()] -- the rule will match all incoming requests.

Note - The Responder will only be used if none of the server instances in the server pool is available.

9. Select **Sorry_Example** in the **response** drop-down list on the **Configuration** tab as shown below.



Responder: Sorry_Example

10. Click **Commit** to save the match rule.

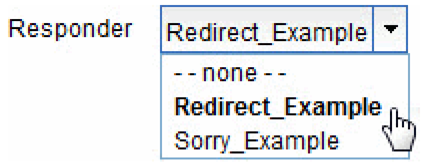
Creating a Match Rule to Redirect All Traffic for a Specific URL

Another common cluster configuration requirement is to be able to automatically redirect all traffic that uses a specific URL. To do this, you need to add a new match rule that:

- matches any incoming request
- has a **Redirect** Responder selected

For example, let’s say that we want all traffic to a cluster that uses the URL `http://cluster/special/` to be redirected to `https://www.example.com/special/`. The following procedure shows you how to add the appropriate Responder and Match Rule:

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Load Balance** configuration tab if it is not already selected.
3. Right click on **Responder** on the left navigational pane and select **AddResponder**. The **Add New Responder** dialog appears.
4. Type **`Redirect_Example`** into the **Name** field and select **Redirect**.
5. Type **`https://www.example.com/special/`** into the **URL** field.
6. Click **Commit** to save the new Responder.
7. Right-click on the name of the cluster for which you want to display the sorry page in the left frame and select **Add Match Rule** from the menu.
8. Refer to "Creating a New Match Rule" on page 445 to configure the match rule. Leave the match rule **expression** set to the default [any()] -- the rule will match all incoming requests. Do *not* select any server pools from the **Server Pool** drop down list. This means that if this match rule’s expression selects a request, the Responder we select will respond to the selected request regardless of the status of the server pools in the cluster.
9. Click **Commit** to create the match rule.
10. Select **Redirect_Example** in the **Responder** drop-down list as shown below.



11. Click **commit** to save the match rule.

After completing the above procedure, all client requests to `http://cluster/special/` will be redirected to `https://www.example.com/special/`, even when all the server instances in a server pool are available.

Responders and Hot Spares

Responders provide functionality that automates the very basic functions of a hot spare server, and off loads them onto Equalizer. If more functionality is desired, than a separate real server should be used as a hot spare for the cluster.

It should also be noted that resources Equalizer uses to service client requests via the Responder feature are resources potentially taken away from processing other client requests. In most cases, Responders might possibly have an effect on performance if all the servers in one or more clusters are down during periods of peak usage.

Responder Statistics and Reporting

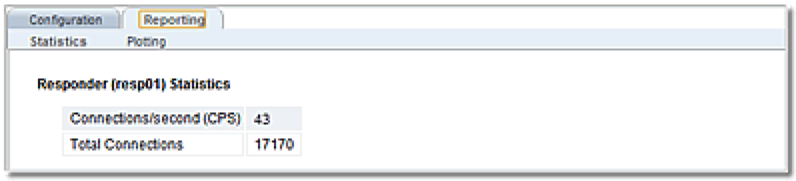
The CLI display of Statistics can be seen by entering the following within the responder context:

Sample Responder Statistics Display

To view the GUI display:

1. Verify that you are logged into the GUI. (Refer to "Logging In" on page 238.)
2. Select the **Load Balance** configuration tab on the left navigational pane if it is not already selected.
3. Click on the arrow (▶) beside **Responders**to expand the branch.
4. Select a Responder and click on the **Reporting** tab to display statistics. The following is an example of the statistics displayed.

Sample Responder GUI Statistics Display



Responder (resp01) Statistics

Connections/second (CPS)	43
Total Connections	17170

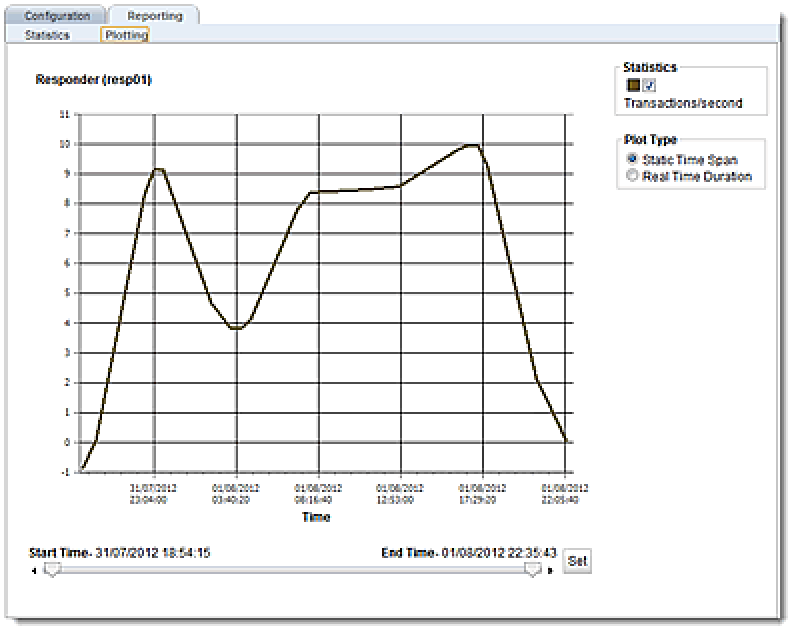
The following are definitions for the statistical terms shown on both the CLI and GUI:

Responder Statistics Definitions

CLI Term	GUI Term	Definition
N/A	Connections/second (CPS)	Connections/second
N/A	Transactions/second (TPS)	Transactions/second
N/A	Throughput	Throughput.
N/A	Total Connections	Total connections.
N/A	Active Connections	Active connections.
N/A	Bytes Received	Bytes received.
N/A	Bytes Sent	Bytes sent.
TOTALPRCSD	N/A	Connections Processed

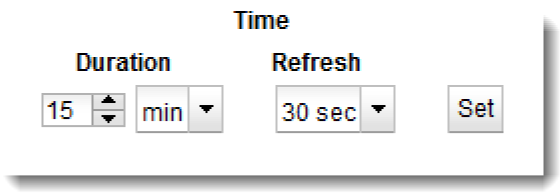
The following is a graphical plot that can be displayed on the GUI. Select a Responder on the left navigational pane and click on the **Reporting** tab and then **Plotting**. The following will be displayed:

Sample Responder Plot



The specific types of statistics that are graphically displayed are determined by the selections on the **Statistics** pane on the upper right corner of the GUI.Make selections based on the data that you require. As you can see from the figure above, the Responder statistic displays only **Transactions/second**.

The **Plot Type** selection determines whether the display shown reflects a Static Time Span which is configured using the slider or whether a real time duration is display. If **Real Time Duration** is selected the slider controls will change to **Duration** and **Refresh** controls as shown below. In this case set the **Duration** of time in which you would like to review statistics and the **Refresh** rate desired.



Chapter 18
Link Load Balancing

Sections in this chapter include:

Link Load Balancing

Multiple ISP connections at each point of presence help to guarantee the availability of your services by building redundancy. Link Load Balancing (LLB) functionality allows your ADC appliance to support multiple upstream links across infrastructure that supports them. If a primary ISP link fails, LLB enables the seamless redirection of traffic through a backup link

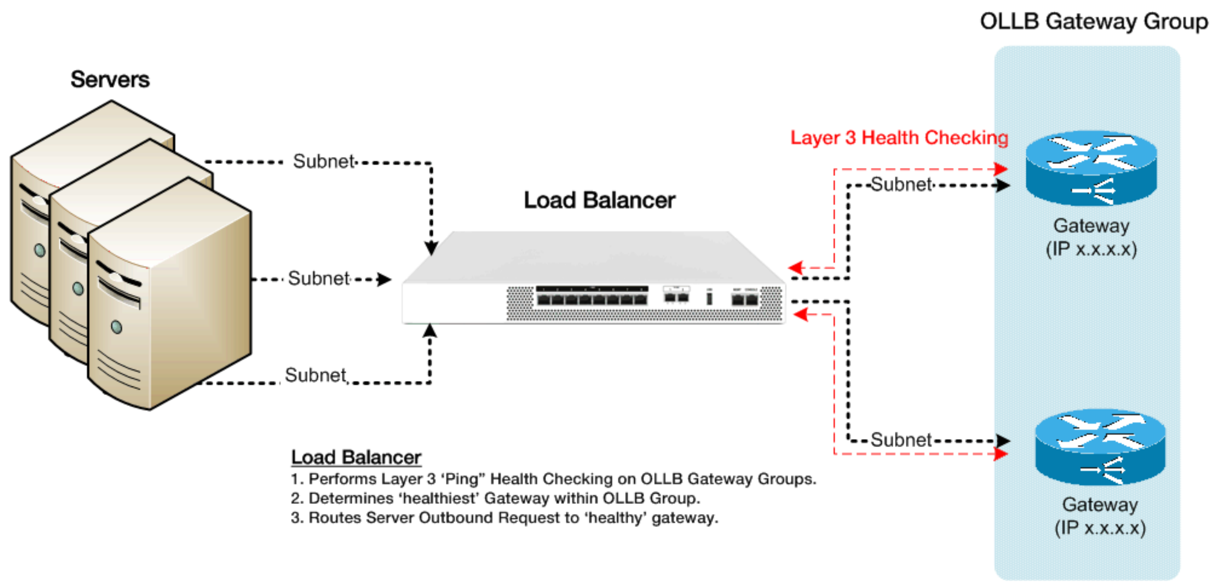
Similar to GSLB, inbound LLB avoids the need for failover via Border Gateway Protocol (BGP) by using DNS-based load balancing and gateways instead. LLB also adds the capability of clients reaching each of your points of presence through multiple paths. It is typically configured for both outbound and inbound traffic.

Outbound Link Load Balancing

Outbound LLB (OLLB) is used when you want redundancy across multiple routes for traffic leaving the ADC. One prerequisite for OLLB is that all gateways configured into an OLLB group must be able to route traffic to the same set of destinations.

In order to distribute outbound traffic from your servers, you must define links by defining its gateway. You will also need to configure one or more health checks that will monitor the availability of the link so that Equalizer can avoid links that are down.

The illustration below shows how OLLB functions:


OLLB Gateway Group
Servers
Layer 3 Health Checking
Subnet
Subnet
Subnet
Subnet
Subnet
Load Balancer
Gateway (IP x.x.x.x)
Gateway (IP x.x.x.x)
Load Balancer
1. Performs Layer 3 'Ping" Health Checking on OLLB Gateway Groups.
2. Determines 'healthiest' Gateway within OLLB Group.
3. Routes Server Outbound Request to 'healthy' gateway.

If you want Equalizer to avoid links that are not available, or links that do not have complete routes to crucial IP addresses, you will need to enable link health checks. Each link health check will periodically send Layer 3 ICMP ECHO probe (pings) from the Equalizer interface IP to an IP address that must be reachable in order for the link to be determined to be available. This can be any IP address, such as a main office, core router, server, cluster or virtual server at another data center.These probes are configured on the gateways (See "Configuring Outbound Link Load Balancing" on page 630 and "Configuring Inbound Link Load Balancing" on page 637).

Configuring Outbound Link Load Balancing

Note - The Link Load Balancing feature supports up to 16 links per OLLB group.

Configuration of OLLB consists of the following:

1. Adding VLANs with subnets
2. Configuring gateways
3. Configuring OLLB groups
4. Configuring NAT
5. Configuring subnet routes

Configuring Outbound Link Load Balancing Using the GUI

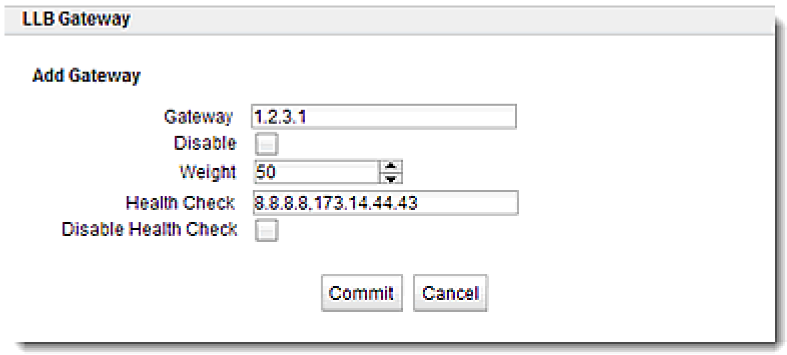
1. Log in to the GUI

Configuring VLANs with Subnets

2. Configure VLANs as described in "Configuring VLANs" on page 304.
3. Add subnets to the VLANs as described in "Configuring Subnets" on page 308. Configure the subnets over which your internal and link traffic and health checking probes will traverse. Enter a name (maximum of 47 characters).
4. Click on the **Link Load Balance** configuration tab on the left navigational pane.

Configuring Gateway

5. Click on the arrow beside **Outbound** to expand the branch and then click on **Gateways** to display the **Link Load Balancing Gateways** display on the right.
6. Click on **+** to display on the LLB Gateway dialogue screen as shown below.



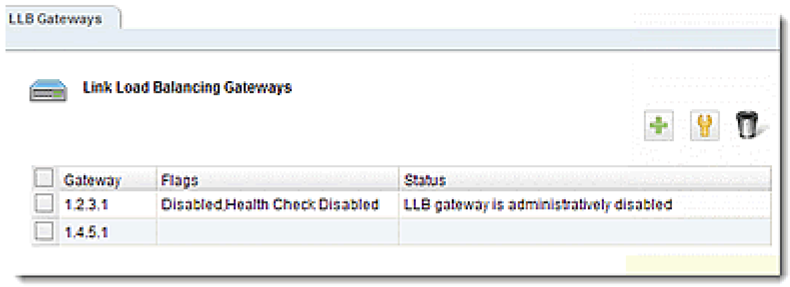
Gateway 1.2.3.1
Disable
Weight 50
Health Check 8.8.8.8.173.14.44.43
Disable Health Check

7. Enter the IP address of the gateway in the **Gateway** text box. Enter the IP addresses of the links to be health checked. Use a "," to separate the IP addresses. You have the option of disabling the Gateway or disabling the Health Check by selecting the appropriate check boxes. By default, the options are not set.

   By default, the **Weight** is set to 50. The weight specified in the gateway object applies to outbound LLB only. When an OLLB route is specified on a subnet, Equalizer will select, from among all that are up, the gateway with the highest weight.

   The **Health Checks** text box allows you to specify a comma-separated list of IP addresses on which a Layer 3 ICMP health check is performed. If one or more of the IP addresses in the list responds to the ICMP health check, then the gateway is considered up; otherwise it is marked down.

8. Click on **Commit** to save the LLB Gateway.
9. Repeat steps 5,6, and 7 for additional LLB Gateways. When the LLB Gateways are configured, they will appear on the list as shown.



Gateway	Flags	Status
1.2.3.1	Disabled,Health Check Disabled	LLB gateway is administratively disabled
1.4.5.1		

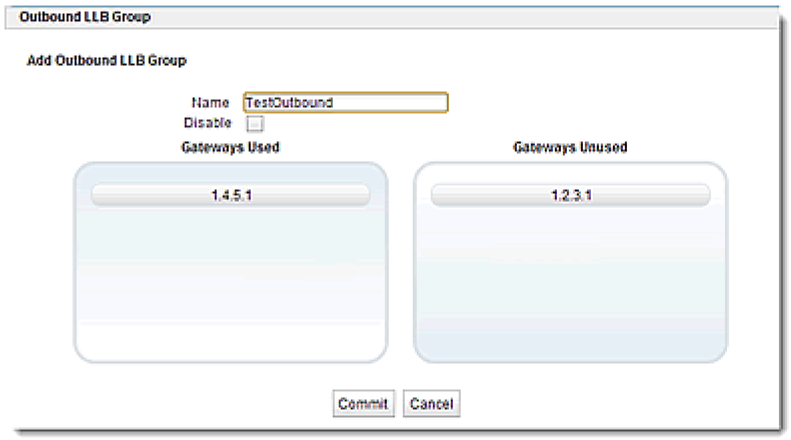
To edit LLB Gateways, either double click a **Gateway** on the list or select a Gateway using the check box and selecting the edit icon. Both methods will display the **LLB Gateway** dialogue screen where changes can be made as necessary.

To delete an LLB Gateway, use the check box to select a Gateway and then click on the trash (delete) icon.

Configure an OLLB Group

10. Click on **Outbound > Groups** from the left navigational pane to display the **Outbound Link Load Balancing Groups** display on the right.
11. Click on **+** to display the **Add Outbound LLB Group** dialogue. An example is shown below.

    The group modification dialogue is the same, although the existing name of the Group will be displayed. Either double click on a group or select a group and then the edit icon to modify an existing group.


Outbound LLB Group

Add Outbound LLB Group

Name: TestOutbound

Disable: ☐

Gateways Used: 1.4.5.1

Gateways Unused: 1.2.3.1

Note - Gateways must have been previously configured.

12. Enter a Name for the group in the space provided.

    You also have the option of disabling the group by selecting the **Disable** check box.

    The **Gateways Used** and **Gateways Not Used** unused panes list all existing Gateways. You can associate one or more with the group by dragging and dropping the gateways.

13. Click on **Commit** to save the OLLB Group.

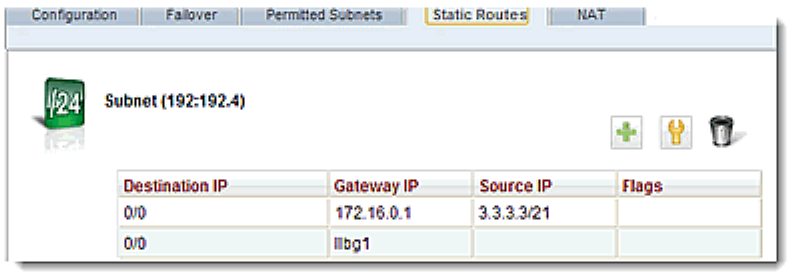
Set Up NAT

14. If needed, set up NAT. Select a subnet from the left navigational pane and click on the **NAT** tab on the right.
15. Click on **+** to activate the Add NAT Rule dialogue.
16. Configure outbound NAT by entering a **from** IP and an **Out** (outbound NAT IP) address.
17. Click on **Commit** to save the NAT rule. The rules will appear on the **Subnet NAT** display.

Set Up Subnet Routes

18. Select the **System** configuration tab if it has not already been selected.
19. Click on the arrow (►) beside Network to expand the branch.
20. Select a VLAN and then select a subnet.
21. Select the **Static Route** tab and click on **+** to add a new route.

22. In the dialogue displayed add a **Destination IP Address** in CIDR format. For example, if you enter an address of 0/0, this would indicate that any destination can be used. Also enter a **Gateway.** The **Gateway** is the packet destination. Enter an LLB Group name that you configured in steps 10-13.
23. Click on **Commit** to save the route. The **Static Route** will be displayed. In the example below, 2 static routes are configured on a subnet 192.4. Both use a **Destination IP** of **0/0.** Note the **llbg1** gateway group name on the second routing entry. This indicates that this subnet (192.4) will be routed outbound through LLB group **llbg1.**



Subnet (192:192.4)

Destination IP	Gateway IP	Source IP	Flags
0/0	172.16.0.1	3.3.3.3/21	
0/0	llbg1		

Configuring Outbound Link Load Balancing Using the CLI

Configuration of OLLB consists of the following:

1. Adding VLANs with subnets
2. Configuring gateways
3. Configuring OLLB groups
4. Configuring NAT
5. Configuring subnet routes

The following is an example of an ILLB configuration using the CLI.

Adding VLANs with Subnets

1. Set up one or more VLANs that will use internal subnets for communication between servers and the ADC, and one or more VLANs that will use external subnets to communicate with the outbound gateways as shown. In the example int is used for the internal VLAN and ext is used for the external VLAN:

```
eqcli> vlan int subnet sn0 ip 1.1.0.2/24
eqcli> vlan int subnet sn1 ip 1.1.1.2/24
eqcli> vlan int subnet sn2 ip 1.1.2.2/24
eqcli> vlan ext subnet sn3 ip 1.2.3.4/24
eqcli> vlan ext subnet sn4 ip 1.4.5.6/24
eqcli> vlan ext subnet sn5 ip 1.6.7.8/24
```

Configuring Gateways

2. Set up LLB Gateways and assign health checks as shown. You can enter up to 16 gateway health checks. By default, the health checks are enabled.

```
eqcli>llb-gw 1.2.3.1 hc 8.8.8.8,173.14.44.43
eqcli>llb-gw 1.4.5.1 hc 8.8.2.2,173.14.32.21
```

3. View the configured gateway using the show command. A message indicating whether the LLB Gateway is enabled is displayed.

```
eqcli > show llb-gw 1.2.3.1
This LLB Gateway is enabled.

LLB Gateway Name        :1.2.3.1
Weight                  :50
Health Check            :8.8.8.8, 173.14.44.43
Flags                   :
```

   a. By default, the gateway’s weight value is 50. If you would like to change the weight, enter the following and enter a value:

```
eqcli > llb-gw 1.2.3.1 weight value
```

b. By default, when you create a gateway it is enabled. If necessary, you can disable the gateway by entering:

```
eqcli > llb-gw 1.2.3.1 flags disable
```

c. By default, LLB gateway health checks are enabled. To disable health checks, enter:

```
eqcli > llb-gw 1.2.3.1 flags disable_hc
```

Configuring OLLB Groups

4. The gateways set up from the previous step are grouped into an OLLB Group (group1) in the example shown.

```
eqcli> ollb-grp group1 gw 1.2.3.1, 1.4.5.1
```

Configuring NAT

5. If needed, set up NAT as shown. Outbound NAT can be set up by entering a *from* parameter in CIDR format that specifies the IP range.

```
eqcli> vlan int subnet sn0 nat from 1.1.0.0/24 out 1.2.3.33 nat sn0 out gw 1.2.3.1
eqcli> vlan int subnet sn0 nat from 1.1.0.0/24 out 1.4.5.33 nat sn0 out gw 1.4.5.1
eqcli> vlan int subnet sn1 nat from 1.1.1.0/24 out 1.2.3.34 nat sn1 out gw 1.2.3.1
eqcli> vlan int subnet sn1 nat from 1.1.1.0/24 out 1.4.5.34 nat sn1 out gw 1.4.5.1
```

Configuring Subnet Routes

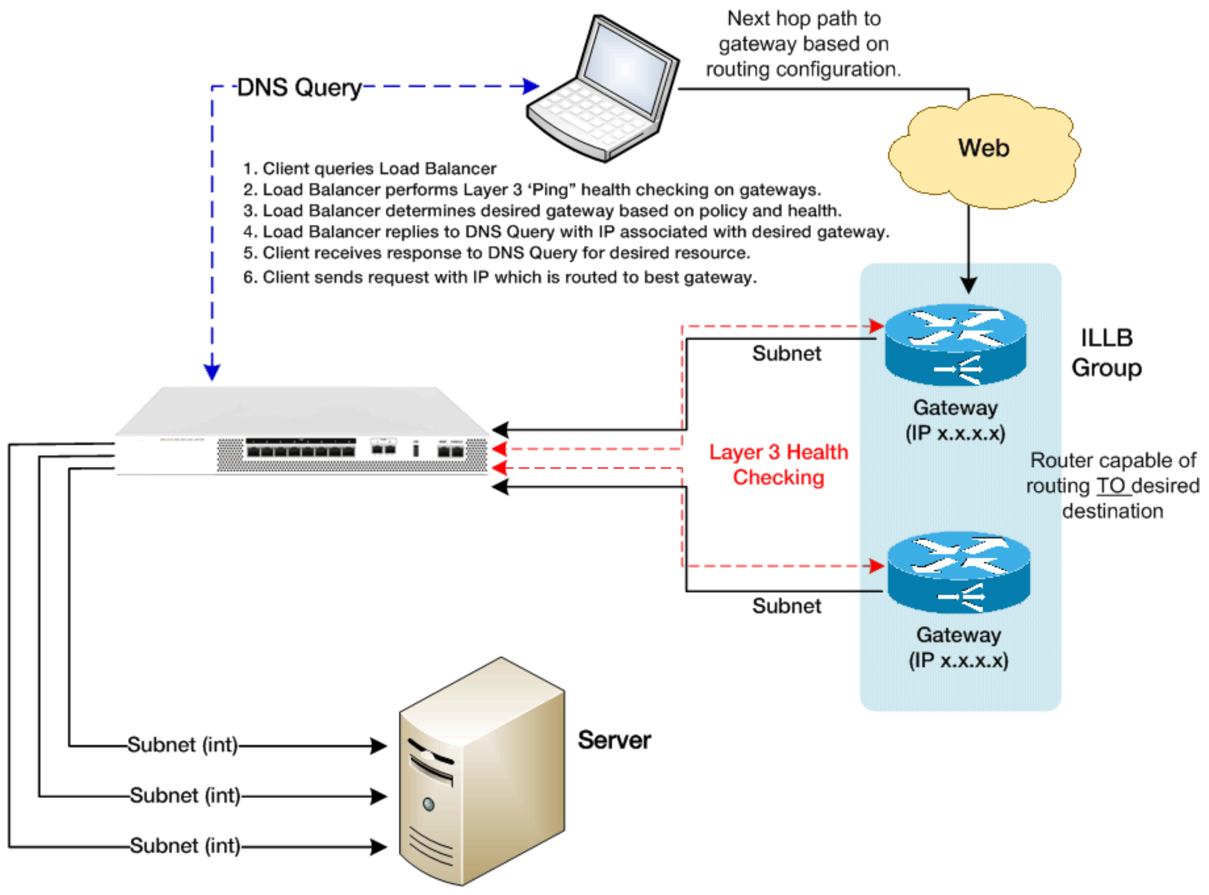
6. In the example shown below subnets sn0 and sn1 will be routed through the outbound LLB group1 (from step 3 above).

```
eqcli> vlan int subnet sn0 route 0/0 gw group1
eqcli> vlan int subnet sn1 route 0/0 gw group1
```

Inbound Link Load Balancing

Similar to GSLB, Inbound Link Load Balancing (ILLB) is DNS-based, in that a client makes a DNS query to resolve the IP address of an FQDN. However, unlike GSLB, the IP address returned in the DNS reply does not represent a geographic location, but rather one of several links available on a single Equalizer.

The illustration below shows how ILLB functions.


DNS Query
Next hop path to gateway based on routing configuration.
Web
1. Client queries Load Balancer
2. Load Balancer performs Layer 3 'Ping" health checking on gateways.
3. Load Balancer determines desired gateway based on policy and health.
4. Load Balancer replies to DNS Query with IP associated with desired gateway.
5. Client receives response to DNS Query for desired resource.
6. Client sends request with IP which is routed to best gateway.
Subnet
ILLB Group
Gateway (IP x.x.x.x)
Layer 3 Health Checking
Router capable of routing TO desired destination
Subnet
Gateway (IP x.x.x.x)
Subnet (int)
Subnet (int)
Subnet (int)
Server

Configuring Inbound Link Load Balancing

In this process, you will be need to specify a Fully Qualified Domain Name (FQDN) that is associated with an ILLB group, and a set of link/IP pairs. The load balancer will, based on the policy in effect (e.g., round robin) and link state, select the appropriate link to use, and return the associated IP address. This IP address may be that of a server or a cluster (or anything else). You must always specify in the configuration that IP which is publicly accessible (e.g., NAT address).

If the ILLB group represents a cluster, in order to have a cluster "presence" on multiple subnets, you must create "copies" of a single cluster such that one exists on each LLB link subnet. The only difference among the clusters will be the IP address. This creates, in effect, a "cluster of clusters". This "cluster of clusters" will be bound together with the ILLB group object.

Configuration of OLLB consists of the following:

1. Adding VLANs with subnets
2. Configuring gateways
3. Configuring ILLB groups
4. Add Targets to the ILLB groups

Using the GUI

1. Log in to the GUI

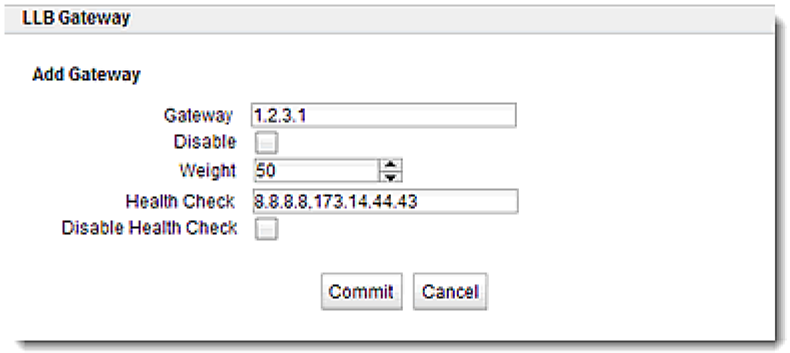
Configuring VLANs with Subnets

2. Configure VLANs as described in "Configuring VLANs" on page 304.
3. Add subnets to the VLANs as described in "Configuring Subnets" on page 308. Configure the subnets over which your internal and link traffic and health checking probes will traverse. Enter a name (maximum of 47 characters)
4. Click on the **Link Load Balance** configuration tab on the left navigational pane.

Configuring Gateways

5. Click on the arrow (►) beside **Outbound** to expand the branch and then click on **Gateways** to display the **Link Load Balancing Gateways** display on the right.

6. Click on + to display on the **LLB Gateway** dialogue screen as shown below.



LLB Gateway

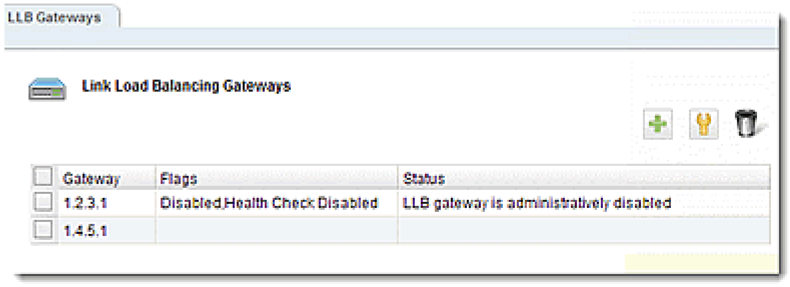
Add Gateway

Gateway	1.2.3.1
Disable	☐
Weight	50
Health Check	8.8.8.8.173.14.44.43
Disable Health Check	☐

7. Enter the IP address of the gateway in the **Gateway** text box. Enter the IP addresses of the links to be health checked. Use a “,” to separate the IP addresses. You have the option of disabling the Gateway or disabling the Health Check by selecting the appropriate check boxes. By default, the options are not set.

   By default, the **Weight** is set to 50. The weight specified in the gateway object applies to outbound LLB only. When an OLLB route is specified on a subnet, Equalizer will select, from among all that are up, the gateway with the highest weight.

8. Click on **Commit** to save the LLB Gateway.
9. Repeat steps 5,6, and 7 for additional LLB Gateways. When the LLB Gateways are configured, they will appear on the list as shown.



Link Load Balancing Gateways

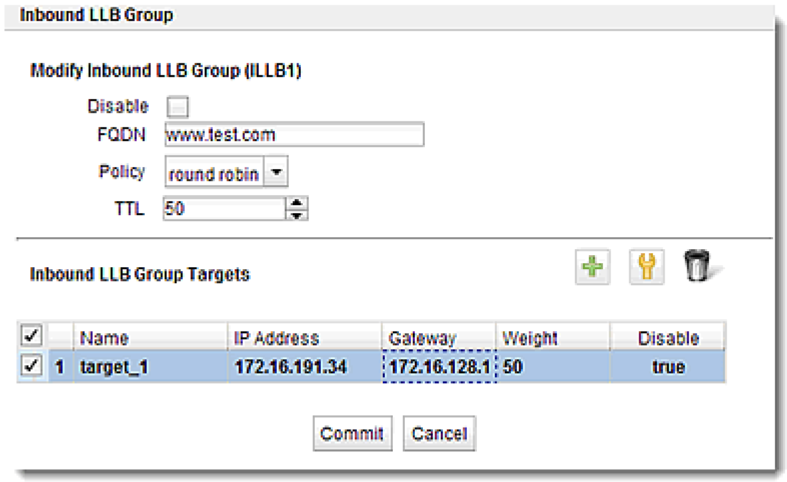
Gateway	Flags	Status
1.2.3.1	Disabled,Health Check Disabled	LLB gateway is administratively disabled
1.4.5.1		

In the list shown above, the Flags that are checked are displayed. .

To edit LLB Gateways, either double click a Gateway on the list or select a Gateway using the check box and selecting the edit icon. Both methods will display the **LLB Gateway** dialogue screen where changes can be made as necessary.

Configuring ILLB Groups

10. The ILLB group is analogous to a geocluster in GSLB. Click on **Inbound > Groups** on the left navigational pane. The **Inbound Link Load Balancing Groups** list will be displayed. Click on **+** to activate the **Add Inbound LLB Group** dialogue as shown below. Note that you can also modify an existing ILLB group by selecting it from the list and either double clicking it or selecting it and clicking the edit icon.



Inbound LLB Group

Modify Inbound LLB Group (ILLB1)

Disable	☐
FQDN	www.test.com
Policy	round robin
TTL	50

Inbound LLB Group Targets

✓		Name	IP Address	Gateway	Weight	Disable
✓	1	target_1	172.16.191.34	172.16.128.1	50	true

Name- a name for the IILLB group

FQDN –Fully Qualified Domain Name. Must include all name components up to the top level (com, net, org, etc). Do not include the trailing period.

TTL - the Time To Live, is the length of time (in seconds) that the client's DNS resolver should cache the resolved IP address. The default is 120 (that is, 2 minutes).

Policy options are:

rr (round robin policy)— causes the load balancer to send a DNS reply with the IP address associated with each available gateway in turn. This is equivalent to traditional "round-robin DNS" load balancing.

Prefer – if more than one gateway is available, this policy selects the gateway with the highest weight and returns the IP associated with that gateway.

flags – by default, the group is enabled

11. Click on **Commit** after entering the ILLB Group information.

Note - You will not be able to enter Target information on this dialogue prior to committing the ILLB Group information.

Add Targets to the OLLB Groups

12. The target describes an IP-gateway pair. If the gateway of the pair is selected as the “best” available gateway according to the policy in effect, the associated IP will be returned in a DNS response. Click on **Inbound > Groups** on the left navigational pane. The **Inbound Link Load Balancing Groups** list will be displayed. Modify an existing ILLB group to add targets by selecting it from the list and either double clicking it or selecting it and clicking the edit icon.
13. Click on **+** to add a new target. By default, the name of the target will be **target_1**, (or **target_2**, etc.) Click in each field to change the **Name**, **IP Address Gateway**, **Weight** and if you would like the target to be enabled (**true**).

Note - A Gateway must have been previously configured.

14. Click on **Commit** to save the target.

Using the CLI

Adding VLANs with Subnets

1. Configure VLANs as described in "Configuring VLANs" on page 304.
2. Configure subnets as described in "Configuring Subnets" on page 308. Configure the subnets over which your internal and link traffic and health checking probes will traverse. An example is shown below:

```
eqcli> vlan int subnet sn0 ip 1.1.0.2/24
eqcli> vlan int subnet sn1 ip 1.1.1.2/24
eqcli> vlan int subnet sn2 ip 1.1.2.2/24
eqcli> vlan ext subnet sn3 ip 1.2.3.4/24
eqcli> vlan ext subnet sn4 ip 1.4.5.6/24
eqcli> vlan ext subnet sn5 ip 1.6.7.8/24
```

Configuring Gateways

3. Set up LLB Gateways and assign health checks. An example shown below. The IP addresses of the health checks are for the IP addresses on which communication to the gateway takes place. You can enter up to 16 gateway health checks. By default the health checks are enabled.

```
eqcli>llb-gw 172.16.128.1 hc 8.8.8.8,173.14.44.43
eqcli>llb-gw 1.4.5.1 hc 8.8.2.2,173.14.32.21
```

4. View the configured gateway by entering the following. A message is displayed, indicating whether the LLB Gateway is enabled/disabled.

```
eqcli > show llb-gw 172.16.128.1
This LLB Gateway is enabled.

LLB Gateway Name   : 172.16.128.1
Weight : 50
Health Check  : 8.8.8.8, 173.14.44.43
Flags :

eqcli >
```

   a. By default, the gateway's weight value is **50**.If you would like to change the weight, enter the following and enter a value:

```
eqcli > llb-gw 172.16.128.1 weight value
```

   b. By default, when you create a gateway the health checks are enabled. If necessary, you can disable the health check by entering:

```
eqcli > llb-gw 172.16.128.1 flags disable_hc
```

   c. To enable the health check after it has been disabled, enter:

```
eqcli > llb-gw 172.16.128.1 flags enable_hc
```

Configuring ILLB Groups

5. The ILLB group is analogous to a geocluster in GSLB. Enter the following:

```
eqcli> illb-grp illb1 fqdn www.test.com ttl 60
```

The **`fqdn`** must include all name components up to the top level (com, net, org, etc). Do not include the trailing period.

The **`ttl`** (Time To Live), is the length of time (in seconds) that the client's DNS server should cache the resolved IP address. The default is 60 (that is, 1 minute).

6. Add a weight and policy to the ILLB group.

```
eqcli> illb-grp illb1 weight 50
```

`weight`-by default, the gateway's weight value is **`50`**. If you would like to change the `weight`, enter the following and enter a value.

```
eqcli > illb-grp illb1 weight value
```

`flags` – by default, the `illb-grp` is enabled. To disable it, enter:

```
eqcli > illb-grp name flags disable
```

7. Display the new `illb-grp` by entering:

```
eqcli > show illb-grp illb1

ILLB Group Name  : name
FQDN : www.test.com
Policy           : rr
TTL              : 60
Flags :

eqcli >
```

Add Targets to the OLLB Groups

8. Add a target to the `illb-grp`. This describes which IP address to return in an A/AAAA record if the specified gateway is selected, or a gateway to consider for the FQDN.

```
eqcli > illb-grp newllb target t1 ip 172.16.191.34 gw 172.16.128.1 weight
50
```

9. Show the new `illb-grp target` by entering:

```
eqcli > show illb-grp newllb target t1

ILLB Target Name   : t1
IP Address : 172.16.191.34
Gateway            : 172.16.128.1
Weight :
Flags :

eqcli >
```

10. Display the list of ILLB group targets by entering:

```
eqcli > show illb-grp illb1 target

Name IP Gateway
t1          172.16.191.34  172.16.128.1

eqcli >
```

Chapter 19

Global Load Balance

Sections in this chapter include:

Overview of Envoy Geographic Load Balancing

Geographic load balancing increases availability by allowing regional server clusters to share workload transparently, maximizing overall resource utilization. The Envoy® Geographic load balancer is an optional software add-on for the Equalizer product line that supports load balancing requests across servers in different physical locations or on different networks.

An Envoy-enabled web site is a geographic server cluster, composed of regional clusters. Each regional cluster is composed of servers that provide a common service, supervised by an Equalizer running Envoy. For example, the web site www.coyotepoint.com might be supported by three regional clusters, located in California, New York City, and London. An Equalizer running Envoy software and web servers with similar content are deployed at each of these locations.

In non-Envoy Equalizer configurations, there is a one-to-one correspondence between a cluster and a website: when a client makes a request for a website (say, www.example.com), the client uses the Domain Name Service (DNS) to resolve the website name to an IP address. For a website that is load balanced by an Equalizer, the IP address returned is the IP address of an Equalizer cluster. After resolving the name, the client sends the request to the cluster IP. When Equalizer receives the client request, it load balances the request across the server pool in the cluster, based on the current load balancing policy and parameters.

In an Envoy conversation, you have two or more Equalizers located in separate locations. Each Equalizer and its set of clusters and servers forms a site (or Envoy site. With Envoy, the website name in the client request is resolved to a GeoCluster IP. A GeoCluster is analogous to a cluster, but one level above it: in other words, a GeoCluster actually points to two or more clusters that are defined on separate Equalizers.

In the same way that Equalizer balances requests for a cluster IP across the server pool in the cluster, Equalizer load balances a request for a GeoCluster IP across the clusters in the GeoCluster configuration. Once a site is chosen and the client request arrives at that site, the request is load balanced across the servers in the appropriate cluster. In this way, you can set up geographically distant Equalizer's to cooperatively load balance client requests.

Envoy on EQ/OS 10 can only interoperate with Envoy running on other units running EQ/OS 10, and does NOT interoperate with Envoy running on EQ/OS 8.6 (or earlier).

Envoy Configuration Summary

Follow this general procedure when setting up Envoy for the first time:

1. Configure appropriate clusters (and servers) on all of the Equalizers to be included as Envoy sites in the GeoCluster.
2. Configure the GeoCluster on each Equalizer; the parameters used should be the same on all sites. This includes creating GeoSites and adding GeoSite Instances to the GeoCluster. (Refer to "Configuring GeoClusters" on page 651 and "Configuring Geosites" on page 656 for details.)
3. Configure the authoritative DNS server for your website’s domain with DNS records for all Equalizers in the GeoCluster. The DNS server returns these records to clients in response to DNS requests to resolve the website (GeoCluster) name.

DNS Configuration

Every Web site is assigned a unique IP address. To access a website, a client needs to know what the site IP address is. Users don't usually enter an IP address into their Web browser, but rather enter a site's domain name instead. In order to access a requested website, a Web browser needs to be able to convert the site's domain name into the corresponding IP address. This is where DNS comes into play.

1. A client computer is configured with the address of a preferred DNS server.
2. A requested URL is forwarded to the DNS server, and the DNS server returns the IP address for the requested website.
3. The client is then able to access the requested site.

Local (Caching) DNS Server

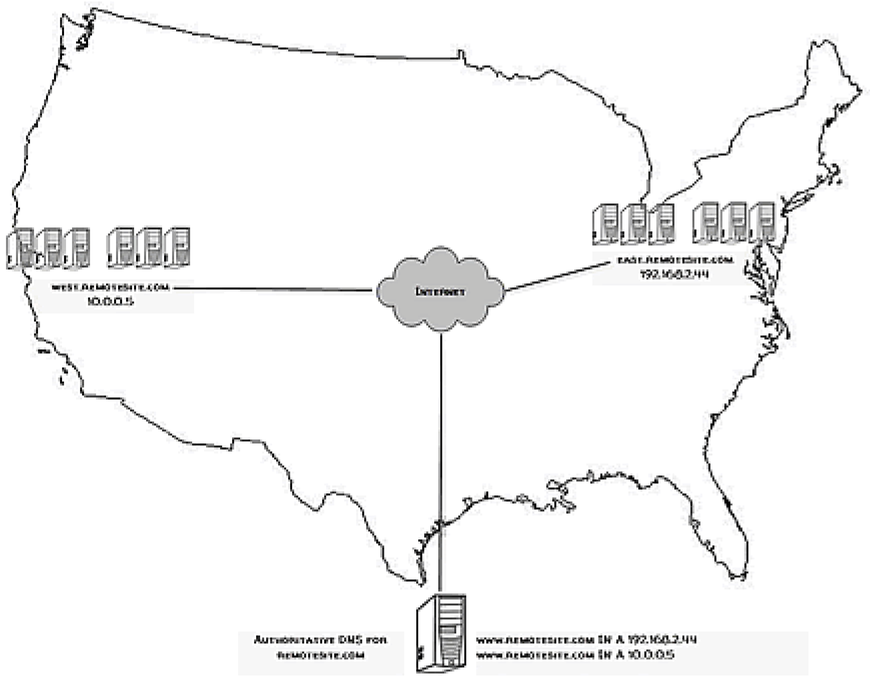
In a typical GSLB configuration a local, or caching DNS server resides in the client's LAN environment. When the client directs the browser to get to a URL (i.e., www.coyotepoint.com) the browser requests the local DNS to resolve the name (i.e. www.coyotepoint.com) to an IP address. Once local DNS server resolves the name to an ip address. It will first check a root name server, which returns a list of name servers for, say, the *.com* domain. The name servers for the domain name space return the IP addresses of an Authoritative DNS server for the domain name. Finally, the Authoritative DNS for the domain name returns the IP address of the web server, or in Envoy SAE, the FQDN of a GeoCluster. For each GeoCluster to be balanced, an Authoritative Name Server must be configured to return name server and alias records for Envoy SAE’s at every regional site

Configuring an Authoritative DNS Name Server for Envoy

You must configure an Authoritative DNS Name Server(s) for the domains that are to be geographically load balanced to delegate authority to the Envoy sites. You need to delegate each of the fully-qualified subdomains to be balanced. If your DNS server is run by an Internet Service Provider (ISP), then you need to ask the ISP to reconfigure the DNS server for Envoy. If you are running your own local DNS server, then you need to update the DNS server’s zone file for your Envoy configuration.

This is usually the last step performed when configuring Envoy. It is recommended to set up Envoy and test your Envoy configuration thoroughly before making changes on the authoritative name server.

For example, assume you must balance www.coyotepoint.com across a GeoCluster containing two Envoy sites, east.coyotepoint.com (at 192.168.2.44) and west.coyotepoint.com (at 10.0.0.5). In this case, you must configure the name servers that will handle the coyotepoint.com domain to delegate authority for www.coyotepoint.com to both east.coyotepoint.com and west.coyotepoint.com. When queried to resolve www.coyotepoint.com, coyotepoint.com's name servers should return name server (NS) and alias (A) address or glue records for both Envoy sites.



An example of a DNS zone file for this configuration is shown below. In this example, the systems ns1 and ns2 are assumed to be the authoritative name servers (master and slave) for the coyotepoint.com domain.

```
$TTL 86400
remotesite.com. IN SOA ns1.remotesite.com.
hostmaster.remotesite.com. (
 0000000000
 00000
 0000
 000000
 00000 )
```

```
remotesite.com. IN NS ns1.remotesite.com.
remotesite.com. IN NS ns2.remotesite.com.
www.remotesite.com. IN NS east.remotesite.com.
www.remotesite.com. IN NS west.remotesite.com.
ns1 IN A ns1-IP-address
ns2 IN A ns2-IP-address
east IN A 192.168.2.44
west IN A 10.0.0.5
```

In the example above, we left the domain parameters as zeros, since these vary widely between DNS installations. Please see the documentation for the version of DNS that you are using for more information on the zone file content and format.

Envoy also supports AAAA (also called "quad-A" records) for IPv6 addresses.

To ensure that you have properly configured DNS for Envoy, you can use the **nslookup** command (supported on most OS platforms) to confirm that the DNS server is returning appropriate records, as in this example:

```
nslookup www.remotesite.com
Server: ns1.remotesite.com
Address: ns1-IP-address

Name: www.remotesite.com
Address: 192.168.2.44
```

Using Envoy with Firewalled Networks

Envoy sites communicate with each other using UDP-based Geographic Query Protocol (GQP). Similarly, Envoy sites communicate with clients using the DNS protocol. If you protect one or more of your Envoy sites with a network firewall, you must configure the firewall to permit the Envoy packets to pass through.

To use Envoy with firewalled networks, you need to configure the firewalls so that the following actions occur:

- Envoy sites communicate with each other on UDP ports 5300 and 5301. The firewall must allow traffic on these ports to pass between Envoy sites.
- Envoy sites and clients can exchange packets on UDP port 53. The firewall must allow traffic on this port to flow freely between an Envoy site and any Internet clients so that clients trying to resolve host names via the Envoy DNS server can exchange packets with the Envoy sites.
- Envoy sites can send ICMP echo request packets out through the firewall and receive ICMP echo response packets from clients outside the firewall. When a client attempts a DNS resolution, Envoy sites send an ICMP echo request (ping) packet to the client and the client might respond with an ICMP echo response packet.

Using Envoy with NAT Devices

If an Envoy site is located behind a device (such as a firewall) that is performing Network Address Translation (NAT) on incoming IP addresses, then you must specify the public (non-translated) IP as the Site IP, and use the translated IP (the non-public IP) as the resource (cluster) IP in the Envoy configuration.

This is because Envoy must return the public cluster IP to a requesting client in order for the client to be able to contact that cluster -- since the request goes through the NAT device before it reaches Equalizer. The NAT device translates the public cluster IP in the request to the non-public cluster IP that is defined on Equalizer, and then forwards the packet to Equalizer.

The non-public cluster IP must still be specified as the resource IP for the site, as this is the IP that Envoy will use internally to probe the availability of the resource (cluster) on the site.

Configuring GeoClusters

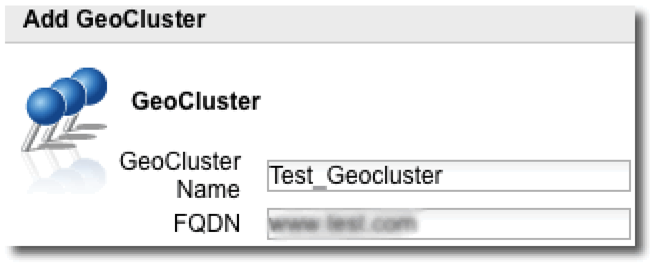
This section shows you how to add or delete a GeoCluster and how to configure a GeoCluster's load-balancing options. Configuring a GeoCluster and its sites is analogous to configuring a virtual cluster and its servers.

There are two parts to configuring GeoClusters. The first is to Add a GeoCluster and the second is to modify the GeoCluster parameters.

Adding a GeoCluster (GUI)

When Envoy is first enabled, there are no GeoClusters defined. To add a GeoCluster:

1. Log in to the GUI(See "Logging In" on page 238).
2. Right click on the **GeoClusters** in the left navigational pane and the **Add GeoCluster** form will be displayed.



Add GeoCluster

GeoCluster

GeoCluster Name	Test_Geocluster
FQDN	www.test.com

3. Enter a **GeoCluster Name** in the space provided.
4. Enter a **FQDN** in the space provided. This is the Fully Qualified Domain Name of the GeoCluster (for example, `www.coyotepoint.com`). The FQDN must include all name components up to the top level (com, net, org, etc). Do not include the trailing period.
5. Click on **Commit** to add the GeoCluster. The new GeoCluster will appear on the left navigational pane.

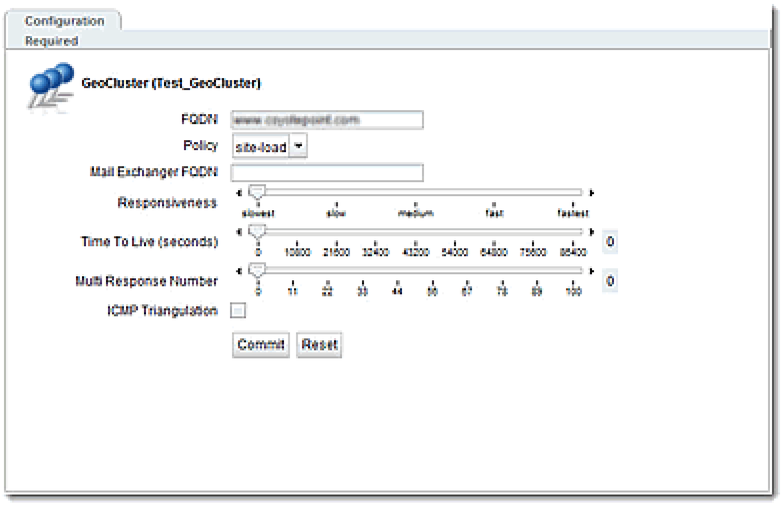
Deleting a GeoCluster (GUI)

1. Log in to the GUI (See "Logging In" on page 238).
2. Right-click on the GeoCluster on the left navigational pane and select **Delete GeoCluster**.

Viewing and Modifying GeoCluster Parameters (GUI)

To view or modify a GeoCluster's load-balancing options, proceed with the following:

1. Log in to the GUI (See "Logging In" on page 238).
2. Click on the GeoCluster on the left navigation pane. The figure below will be displayed:



Configuration Required

GeoCluster (Test_GeoCluster)

FQDN: www.coyotepoint.com

Policy: site-load

Mail Exchanger FQDN:

Responsiveness: slowest

Time To Live (seconds): 0

Multi Response Number: 0

ICMP Triangulation: ☐

3. View and Modify paramaters using the following guidelines:

FQDN	The **GeoCluster name** which is the fully-qualified domain name (FQDN) of the GeoCluster (for example, `www.coyotepoint.com`). The FQDN must include all name components up to the top level (com, net, org, etc). Do not include the trailing period

Policy	Three basic metrics are used by the policy to load balance requests among sites: the current load on the site, the initial weight setting of the site, and ICMP triangulation responses. The policy setting tells Envoy the realtive weight to assign to each metric when choosing a site: **round robin** causes Envoy to send requests to each available site, in turn, in the order they are listed in the configuration. This is equivalent to traditional ‘round-robin DNS’ load balancing. **round trip** weights the ICMP triangulation information received from each site more heavily than other criteria. **adaptive** give roughly equal weights to the site load and ICMP triangulation responses, and gives less weight to the initial weight for the site. This is the default setting. **site load** weights the current load at each site more heavily than other criteria. **site weight** weights the user-defined initial weight for each site more heavily than other criteria.
Mail Exchanger FQDN	The fully qualified domain name (e.g., "mail.example.com") to be returned if Equalizer receives a “mail exchanger” request for this GeoCluster. The mail exchanger is the host responsible for handling email sent to users in the domain. This field is not required.
Responsiveness	This value controls how aggressively Equalizeradjusts the site’s dynamic weights. Equalizer provides five response settings: **slowest, slow, medium, fast**, and **fastest**. Faster settings enable Equalizer to adjust its load balancing criteria more frequently and permit a greater variance in the relative weights assigned to sites. Slower settings cause site measurements to be averaged over a longer period of time before Equalizer applies them to the cluster-wide load balancing; slower settings also tend to ignore spikes in cluster measurements caused by intermittent network glitches. Use the slider to make your selection. We recommend that you select the medium setting as a starting point.
Time To Live	The cache time-to-live, which is the length of time (in seconds) that the client’s DNS server should cache the resolved IP address. Longer times will result in increased failover times in the event of a site failure, but are more efficient in terms of network Resources. Use the slider to make your selection. The default is 120 (that is, 2 minutes).
Multi Response Number	This is the maximum number of Resource records returned in a DNS response that will be allowed in this GeoCluster. The first address will be the actual selected GeoSite. Those that follow will be any site which is up in the list of GeoSites.

ICMP triangulation (option)	When a request for name resolution is received by Envoy from a client's local DNS, this option (if enabled) tells Envoy to request network latency information from all sites in order to make load balancing decisions based on the proximity of each site to the client's DNS server. To do this, all Envoy sites send an ICMP echo request ("ping") to the client's DNS server. The reply from the DNS server allows Equalizer to select a site using the length of time is takes for the DNS server's reply to reach the site. (Consequently, this method assumes that the client's DNS server is geographically close to the client -- which is usually the case.) If you do not want Envoy GeoSites to ping client DNS servers, disable this flag (this is the default setting). Please note that in order for ICMP triangulation data to be collected at each GeoSite: The client's DNS server must be configured to respond to ICMPv4 echo requests (ICMPv6 is not currently supported for Envoy triangulation). The client's DNS server must be allowed to respond through any firewalls between it and the Envoy GeoSites.

Note - For all policies, the current site load metric is ignored for the first 10 minutes that the site is up, so that the metric value is a meaningful measure of the site load before it is used.

Note - In Version 10, if ICMP triangulation is enabled and all GeoSites report that triangulation failed, then ICMP triangulation is ignored for GeoSite selection. That is, Envoy geographic load balancing will proceed as if ICMP triangulation were disabled. [In Version 8.6, if no GeoSites successfully completed ICMP triangulation then Envoy would send a NULL response.]

If only some GeoSites report failed triangulation, and there are others that did not fail and that are not down, then GeoSite selection will only include those sites that successfully completed ICMP triangualtion. [This is the same as the Version 8.6 behavior.]

Adding a GeoCluster (CLI)

To add a GeoCluster using eqcli as follows:

1. Log in to eqcli as described in Starting the CLI.
2. Enter the following at the CLI prompt:

```
eqcli > geocluster gcname req_cmds
```

Deleting a GeoCluster (CLI)

Delete a GeoCluster using eqcli as follows:

1. Log in to eqcli as described in Starting the CLI.
2. Enter the following at the CLI prompt:

```
eqcli > no geocluster gcname
```

Viewing and Modifying GeoCluster Parameters (CLI)

To add a GeoCluster using eqcli as follows:

1. Log in to eqcli as described in Starting the CLI.
2. Use the parameter descriptions above and the command line sequences described in GeoCluster and GeoSite Instance Commands to view and modify GeoCluster parameters using eqcli.

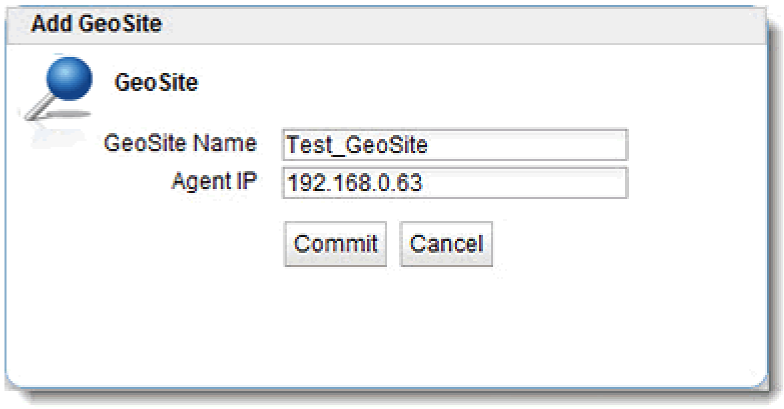
Configuring Geosites

In EQ/OS 10, GeoSites are defined separately (like Servers) and then added to GeoClusters as GeoSite Instances. This section describes how to add, delete and configure GeoSites and includes descriptions of the parameters used by GeoSites.

Adding a GeoSite (GUI)

To add a GeoSite using the GUI proceed with the following:

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Global Load Balance** configuration tab on the left navigational pane if it is not already selected.
3. Right-click on **GeoSites** and select **Add GeoSite**. The **Add GeoSite** form will be displayed.
4. Enter a **GeoSite Name** and an **Agent IP**. The **Agent IP** is the IP address of the GeoSite's Envoy Agent. This is the subnet IP address of Envoy at this site on which the Envoy Agent is running. On EQ/OS 10, you can enable the Envoy Agent on any subnet's VLAN IP address or Failover IP address.



GeoSite Name: Test_GeoSite
Agent IP: 192.168.0.63

5. Click on **Commit** to add the GeoSite.The new GeoSite will appear on the left navigational pane .You can view the GeoSite configuration by clicking on the GeoSite in the left navigational pane which will display the **GeoSite > Configuration > Required** screen on the right.

Deleting a GeoSite (GUI)

To delete a GeoSite using the GUI proceed with the following:

1. Log in to the GUI (See "Logging In" on page 238).
2. Right-click on a GeoSite on the left navigational pane and select **Delete GeoSite.**

Adding a GeoSite (CLI)

Too add a GeoSite using eqcli as follows:

1. Log in to eqcli as described in "Starting the CLI" on page 142.
2. Enter the following at the CLI prompt:

```
eqcli > GeoSite gsnamereq_cmds
```

Deleting a GeoSite (CLI)

Too delete a GeoSite using eqcli as follows:

Log in to eqcli as described in "Starting the CLI" on page 142.

Enter the following at the CLI prompt:

```
eqcli > no GeoSite gsname
```

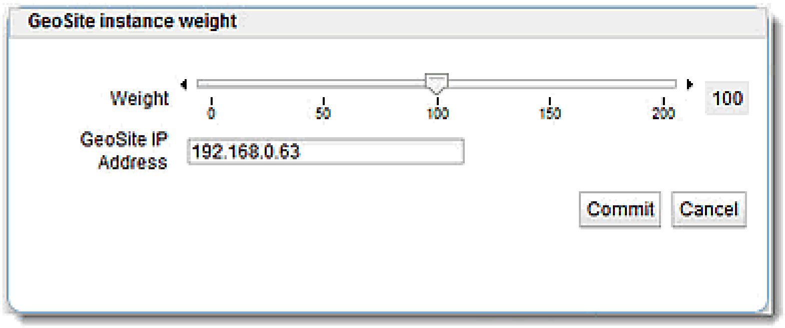
Geosite Instance Parameters

The following procedures describe the process of adding and configuring GeoSite Instance parameters.

Adding and Configuring a GeoSite Instance (GUI)

To add a GeoSite instance to a GeoCluster using the GUI proceed with the following:

1. Log in to the GUI (See "Logging In" on page 238).
2. Refer to "Configuring GeoClusters" on page 651 to configure a GeoCluster.
3. Add a GeoSite instance to a GeoCluster using one of the following methods:
   a. Using the GUI drag and drop functionality, click on a GeoSite on the left navigational pane and drag it to the desired GeoCluster on the tree. The **GeoSite instance weight** form will be displayed.

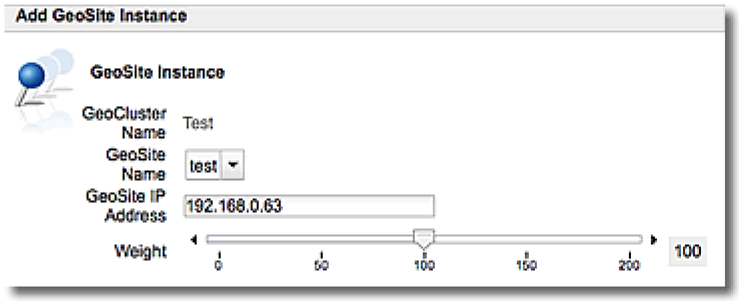


GeoSite instance weight

Weight: 100

GeoSite IP Address: 192.168.0.63

   b. Right click on a GeoCluster on the left navigational pane and select Add GeoSite Instance. The Add GeoSite Instance form will be displayed. If this method is used, you will need to enter the **GeoSite IP Address** and select the desired GeoSite using the **GeoSite Name** drop down list.



Add GeoSite Instance

GeoSite Instance

GeoCluster Name: Test

GeoSite Name: test

GeoSite IP Address: 192.168.0.63

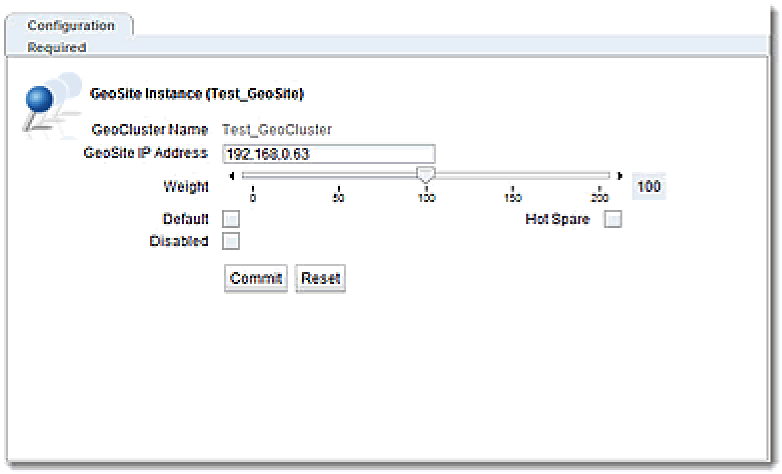
Weight: 100

4. In both methods of creating GeoSite Instances the **GeoSite IP Address** is required. This is the IP address returned by DNS to a client when the GeoCluster is accessed. For example, when a client opens `www.coyotepoint.com`, the local DNS server returns an A record that contains the IP address for `www.coyotepoint.com`. This is usually the address of an Equalizer cluster and in this case is also used as the resource IP. However, the site's A record IP may be different from the cluster (resource) IP if the A record IP address is NAT'ed to an internal address (the actual cluster IP). In this case, you specify the A record IP as the site IP and the cluster IP as the resource IP.
5. Using the slider on either form, adjust the **Weight**, which is an integer that represents the site's capacity. (This value is similar to a server's initial weight.) Valid values range between 10 and 200. Use the default of 100 if all sites are configured similarly; otherwise, adjust higher or lower for sites that have more or less capacity.

   Equalizer uses a site's initial weight as the starting point for determining what percentage of requests to route to that site. Equalizer assigns sites with a higher initial weight a higher percentage of the load. The relative values of site initial weights are more important than the actual values. For example, if two sites are in a GeoCluster and one has roughly twice the capacity of the other, setting the initial weights to 50 and 100 is equivalent to setting the initial weights to 100 and 200.

   Dynamic site weights can vary from 50% to 150% of the assigned initial weights. To optimize GeoCluster performance, you might need to adjust the initial weights of the sites in the cluster based on their performance.

   Site weights can range from 10 to 200. When you set up sites in a GeoCluster, you should set each site's initial weight value in proportion to its capacity for handling requests. It is not necessary for all of the initial weights in a cluster to add up to any particular number.
6. Click on **Commit** to add the GeoSite instance. The instance will appear beneath the GeoCluster on the left navigational pane .
7. Select additional options for the GeoSite instance by clicking the GeoSite instance from the left navigational pane to display the **Configuration > Required** screen.From this screen all configuration options can be modified.



GeoSite Instance (Test_GeoSite)

GeoCluster Name Test_GeoCluster
GeoSite IP Address 192.168.0.63
Weight 100
Default
Hot Spare
Disabled

- **Default -** Designates this site as the default site for the GeoCluster. Envoy load balances to the default site whenever it cannot choose a site based on the GQP probe information it gets from the sites. This can happen, for example, when GQP probe responses are not received from any site, when the resource (cluster) is down at all available sites, etc. If no default site is selected for a GeoCluster and all sites are down, then Envoy sends a null response to the client DNS.
- **Hot Spare -** When enabled, no traffic will be routed to the site unless there are no other sites available. Default value is disabled.
- **Disabled -** When turned on, no traffic will be routed to the site. Default value is off.

Deleting a GeoSite Instance (GUI)

To remove a GeoSite instance from a GeoCluster using the GUI proceed with the following:

1. Log in to the GUI (See "Logging In" on page 238).
2. Click on the GeoSite Instance on a GeoCluster branch on the left navigational pane and select **Delete GeoSite Instance**.

Adding and Configuring a GeoSite Instance (CLI)

To add and configure a GeoSite instance using eqcli proceed with the following:

1. Log in to eqcli as described in "Starting the CLI" on page 142
2. Use the parameter descriptions above and the command line sequences described in "GeoSite and GeoSite Resource Commands" on page 187 to view and modify GeoSite parameters using eqcli.

Deleting a GeoSite Instance (CLI)

To remove a GeoSite instance from a GeoCluster using eqcli proceed with the following:

1. Log in to eqcli as described in "Starting the CLI" on page 142.
2. Enter the following at the CLI prompt:

```
eqcli > no geocluster   gclname gsi gsimaname
```

where:

gclname is the name of the GeoCluster

gsi is the GeoSite instance

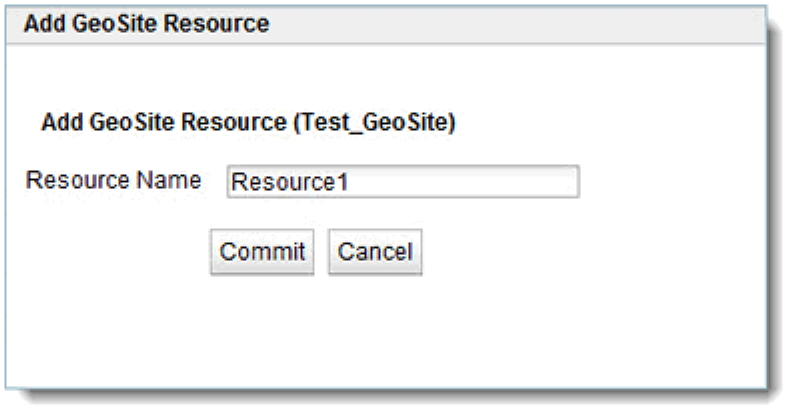
gsimaname is the name of the GeoSite instance.

GeoSite Resources and GeoSite Instance Resources

GeoSite Resources are named clusters defined within a GeoSite. They are assigned a name so that they can be configured into a GeoCluster. For example a GeoSite in New York may have a cluster “CLNY1” defined. Another GeoSite located in San Jose may have a cluster “CLSJ1” defined. In order to load balance between the New York and San Jose GeoSites the Resources of each will be defined as GeoSite Resource Instances in a GeoCluster.

Name a GeoSite Resource (GUI)

1. Log in to the GUI (See "Logging In" on page 238).
2. Select a GeoSite from the left navigational pane.
3. Right-click on the GeoSite and select **Add GeoSite Resource** and the following will be displayed.


Add GeoSite Resource

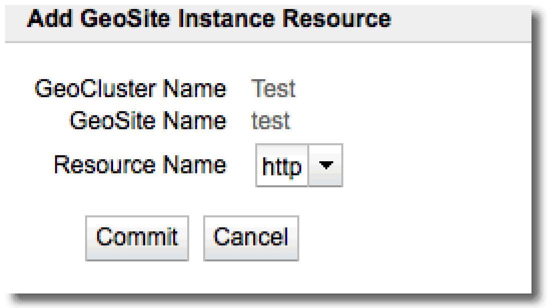
Add GeoSite Resource (Test_GeoSite)

Resource Name: Resource1

4. Enter a name for the Resource and click on **Commit**. The GeoSite Resource will appear on the left navigation pane.

Add a GeoSite Instance Resource to a GeoCluster (GUI)

1. Log in to the GUI (See "Logging In" on page 238).
2. Right click the GeoSite Instance within a GeoCluster on the left navigation pane an select **Add GeoSite Instance Resource.** The following will be displayed:


Add GeoSite Instance Resource

GeoCluster Name: Test
GeoSite Name: test
Resource Name: http

3. Use the **Resource Name** drop down list to select one of the previously defined GeoSite Resources.

4. Click on Commit to add the Resource Instance. It will be displayed on the left navigation tree .

Name a GeoSite Resource (CLI)

1. Log in to eqcli as described in "Starting the CLI" on page 142.
2. Enter the GeoSite context and add the following at the CLI prompt:

```
eqcli ga-gsname> resource clname
```

where:

clname is the cluster name at the GeoSite.

Add a GeoSite Resource Instance to a GeoCluster (CLI)

1. Log in to eqcli as described in "Starting the CLI" on page 142.
2. Enter the GeoCluster context and following at the CLI prompt:

```
eqcli > gcl-gclname> GeoSite gsname resource clname
```

where:
`gcl` is the GeoCluster name
`gsname` is the GeoSite name.
`clname` is the name of the GeoSite Resource

FortiDirector™ GSLB

FortiDirector™ is a Global Service Load Balancing (GSLB) service that provides a cloud-based alternative to the native Envoy GSLB service installed with Equalizer. FortiDirector is completely cloud-based and requires no additional software to be installed on monitored devices. It takes inbound queries and responds with DNS or HTTP redirect responses that are determined by user-defined health checks. These health checks specify how each network resource is to be monitored, and what the thresholds are for considering the network resource to be “UP” or “DOWN” for the purpose of making GSLB decisions.

The Equalizer GUI includes an interface to FortiDirector that allows you to:

- Sign up for a free FortiDirector trial account. As part of the sign up process, a network resource for the ADC on which you sign up is automatically created within FortiDirector.
- Monitor HTTP and DNS redirect statistics generated by the FortiDirector service for devices connected to your account.

Other activities associated with FortiDirector must be performed on the FortiDirector website. These include:

- Setting up health checks for monitoring the network resource associated with your free trial account.
- Creating additional network resources.
- Managing your FortiDirector account.

For additional information about FortiDirector, follow this link to the FortiDirector Documentation Page which provides a full documentation set and videos about the service.

Using the Equalizer GUI to Interface with FortiDirector

The following sections describe:

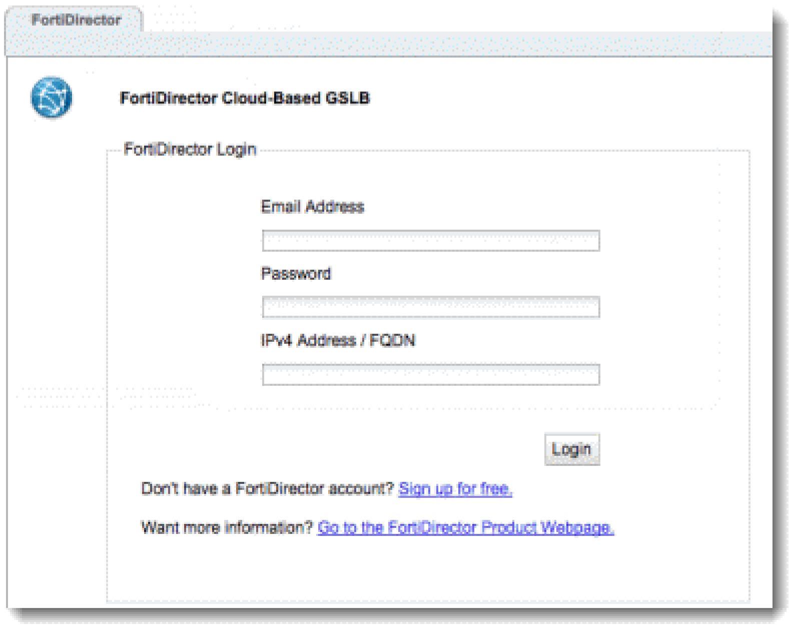
- Creating and Activating a Free FortiDirector Account using the GUI.
- Logging in to an Existing FortiDirector Account and Registering a Network Resource using the GUI.
- Viewing FortiDirector Account Statistics.

Note - DNS must be configured on Equalizer prior to interfacing with FortiDirector. Refer to "Parameters" on page 275 for descriptions.

Creating and Activating a Free FortiDirector Account using the GUI

Follow this procedure if you want to create and activate a free FortiDirector Account.

1. Select **Global Load Balance > FortiDirector GSLB > Log In** on the left navigational pane to display the **FortiDirector Login** screen as shown below. You should note that the **FortiDirector Login** screen is only displayed when your Equalizer is not registered with FortiDirector.


FortiDirector

FortiDirector Cloud-Based GSLB

FortiDirector Login

Email Address

Password

IPv4 Address / FQDN

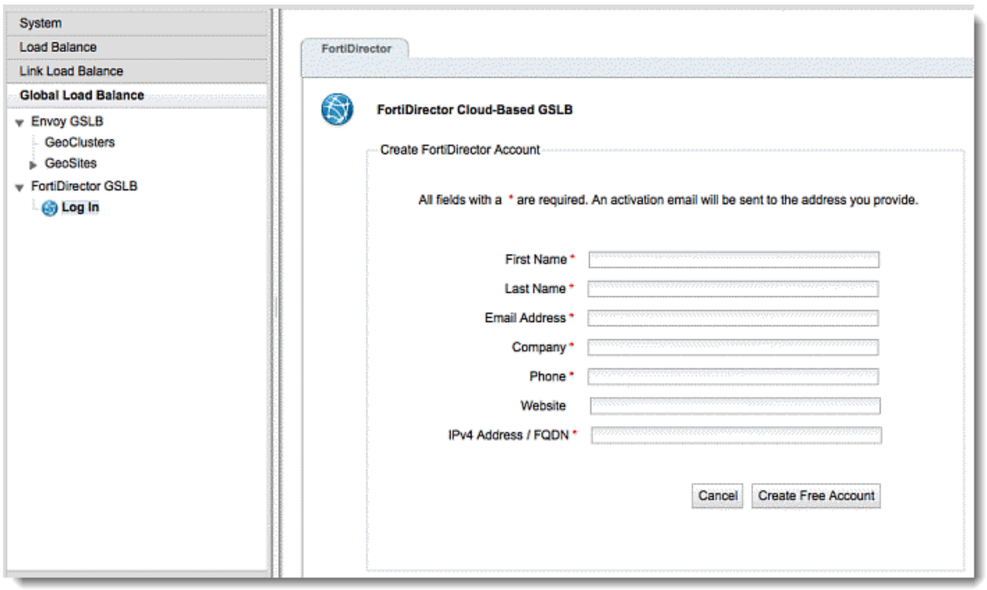
Login

Don't have a FortiDirector account? Sign up for free.

Want more information? Go to the FortiDirector Product Webpage.

2. Click on the "Sign up for free" hyperlink on the login page. This will display the **Create FortiDirector Account** page as shown below. The sign up procedure creates the association between your ADC and FortiDirector. The information required for activation are **First Name**, **Last Name**, **Email Address**, **Company** and **Phone**. **Website** can be your company's URL. This is an optional, information field that has no effect on the functionality of FortiDirector.

IPv4 Address / FQDN information is required to create a network resource on FortiDirector that is capable of servicing user requests. This is the IP address of the Equalizer cluster that FortiDirector will monitor. It can also be an external IP which NATS to a cluster IP.



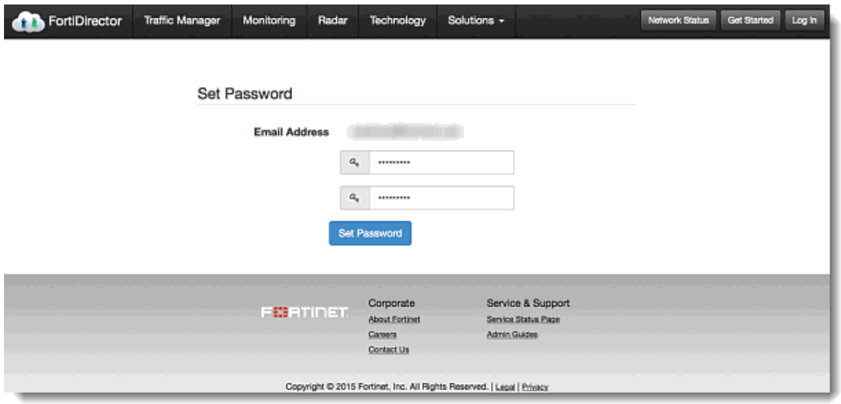
FortiDirector Cloud-Based GSLB

Create FortiDirector Account

All fields with a * are required. An activation email will be sent to the address you provide.

First Name *	
Last Name *	
Email Address *	
Company *	
Phone *	
Website	
IPv4 Address / FQDN *	

3. Click on the **Create Free Account** button after entering the required information. The **FortiDirector login** screen will be redisplayed and you will receive a confirmation email to the address that you provided.
4. When you receive the confirmation email, follow the link in the email to complete the sign up process and set a **Password**. This is the password that you will use to access FortiDirector statistics through the Equalizer GUI, or to log in directly on the FortiDirector website. After entering a password and clicking on the **Set Password** button, you will be prompted to log in to FortiDirector.
5. When you receive the confirmation email, follow the link in the email to complete the sign up process and set a **Password**. This is the password that you will use to access FortiDirector statistics through the Equalizer GUI or to log in directly on the FortiDirector website. After entering a password and clicking on the **Set Password** button, you will be prompted to log in to FortiDirector.



Email Address

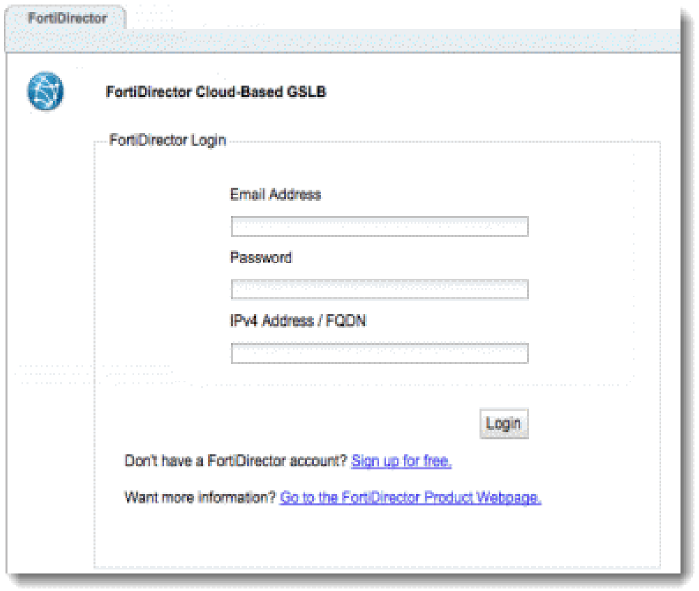
Copyright © 2015 Fortinet, Inc. All Rights Reserved. | Legal | Privacy

6. After you log in, the FortiDirector dashboard will be displayed. Click on the **Network Resources** tab at the top of the FortiDirector Dashboard. Note that a new **DNS Network Resource** has been added, based on the information that you provided on the **Create FortiDirector Account** page.
7. Configure health checks for monitoring the network resource associated with your free trial account. Refer to the FortiDirector Documentation Page to view documentation that describes the health check configuration process.
8. Note that the **Log In** icon on the left navigational pane is changed to **Usage Information.** Each time that you click on the **Usage Information** icon, statistical information will be downloaded from FortiDirector and you will be able to view DNS or HTTP redirect statistics.

Logging in to an Existing FortiDirector Account and Registering a Network Resource using the GUI

Use this procedure if you have an existing FortiDirector account with resources that have been configured with health checks. Refer to the FortiDirector Documentation Page to view documentation that describes the health check configuration process.

1. Select **Global Load Balance > FortiDirector GSLB > Log In** on the left navigational pane to display the **FortiDirector Login** screen as shown below.



Don't have a FortiDirector account? Sign up for free.

Want more information? Go to the FortiDirector Product Webpage.

2. Enter your **Email Address** and **Password** in the space provided and click on the **Login** button. These are the log in credentials that you obtained from previously subscribing directly on the FortiDirector website .
3. Enter an **IPv4 Address/FQDN** in the space provided. This information is required to associate an existing network resource on FortiDirector that is capable of servicing user requests. This is the IP address of the Equalizer cluster that FortiDirector will monitor. It can also be an external IP which NATS to a cluster IP.

4. Click on the **Login** button. The **Log In** icon on the left navigational pane is changed to **Usage Information**. Each time that you click on the **Usage Information** icon, statistical information will be downloaded from FortiDirector and you will be able to view DNS or HTTP redirect statistics.

Viewing FortiDirector Account Statistics.

If a network resource does not have health checks associated with it, statistics will *not* be available. You will need to configure health checks on the network resource. You must specify how each network resource is to be monitored, and what the thresholds are for considering the network resource to be "UP" or "DOWN" for the purposes of inclusion or exclusion from load balancing. Follow the procedures described in the documentation provided on the FortiDirector Documentation Page.

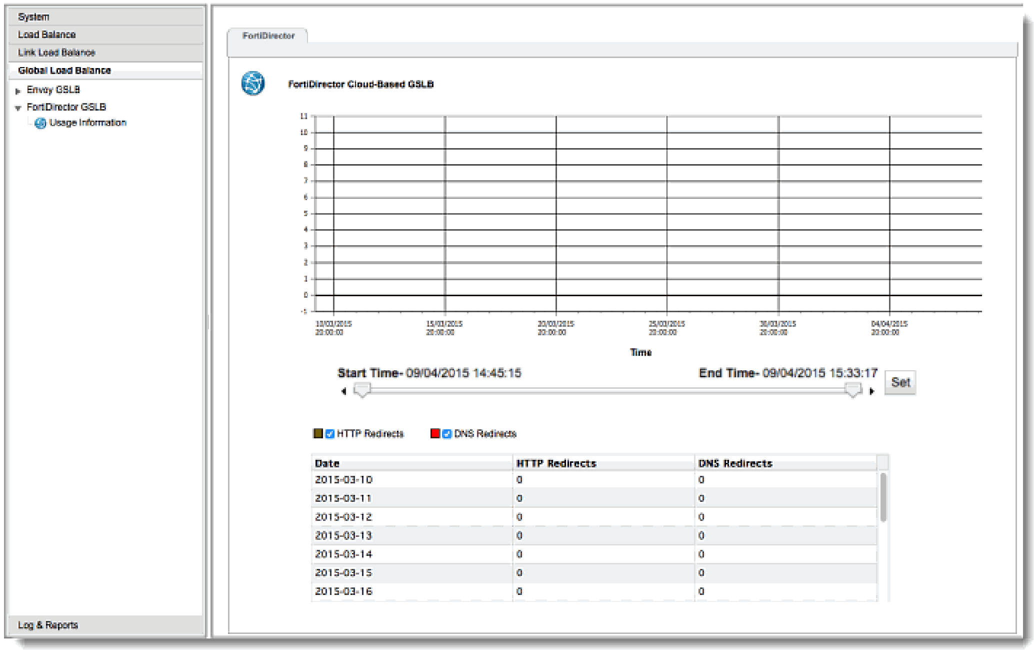
1. Register your Equalizer and create network resources on FortiDirector by either:

   a. Logging in to an Existing FortiDirector Account and Registering a Network Resource using the GUI

   b. Creating and Activating a Free FortiDirector Account using the GUI

2. Configure the network resources with health checks on the FortiDirector website.

3. Display DNS Redirect and HTTP Redirect statistics by selecting **Global Load Balance > FortiDirector GSLB > Usage Information** on the left navigational pane. The following is an example of the display. Each time that you click in the icon, the information on the right pane will be refreshed.



Date	HTTP Redirects	DNS Redirects
2015-03-10	0	0
2015-03-11	0	0
2015-03-12	0	0
2015-03-13	0	0
2015-03-14	0	0
2015-03-15	0	0
2015-03-16	0	0

A slider, beneath the plot allows your set a time frame to monitor HTTP and DNS Redirect statistics. Click on **Set** when you move the slider.

Chapter 20

Logs and Reports

Sections within this chapter include:

Displaying Logs

Logs can be displayed in both the CLI and the GUI.

In the CLI, use the following command:

```
eqcli > show log log type number of lines range datetime1-datetime2
```

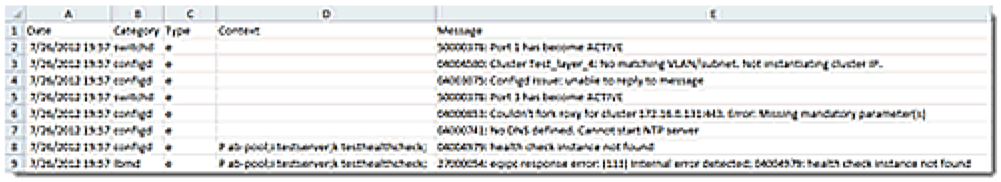
Substitute **`log type`** as **`eq`**, **`sys`**, or **`audit`** to display the event log; system log, or audit log, respectively. B default, the entire log is displayed. Use the **`range`** to specify the time frame of log entries to display.

In the GUI:

1. Click on the **Log and Reports** configuration tab in the left navigational pane.
2. Click on the arrow (▶) beside **Logging** to expand the branch.
3. Click one either **Events Log**, **System Log**, **Audit Log**, **Upgrade Log**, or **Remote Syslog** to display the graphical log browser.

Export to CSV

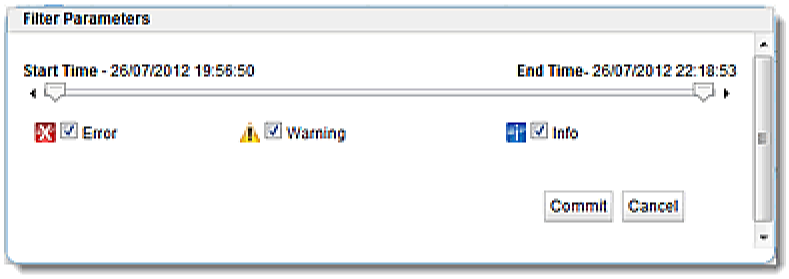
Click on the **Export to CSV** button to download the load in comma separated values (*.csv) format. The file name will be in the format **Equalizer-mon-dd[time frame]EventLog.csv.** An example is shown below. This is an example of a change added to this document.



	A	B	C	D	E
1	Date	Category	Type	Context	Message
2	3/26/2012 19:57	switchd	e		50000378: Port 1 has become ACTIVE
3	3/26/2012 19:57	configd	e		04004580: Cluster Test_layer_4: No matching VLAN/subnet. Not instantiating cluster IP.
4	3/26/2012 19:57	configd	e		04003075: Configd issue: unable to reply to message
5	3/26/2012 19:57	switchd	e		50000378: Port 3 has become ACTIVE
6	3/26/2012 19:57	configd	e		04000813: Couldn't fork proxy for cluster 172.16.0.131:443. Error: Missing mandatory parameter(s)
7	3/26/2012 19:57	configd	e		04000741: No DNS defined. Cannot start NTP server
8	3/26/2012 19:57	configd	e	P ab-pool;s testserver;k testhealthcheck;	04004979: health check instance not found
9	3/26/2012 19:57	lbmd	e	P ab-pool;s testserver;k testhealthcheck;	27000054: egipc response error: [111] Internal error detected: 04004979: health check instance not found

Filtering Status Details

After displaying events for all of Equalizer's configured objects or individual objects, the events displayed in the table can be filtered by specifying **Start Times** and **End Times.** Click on **Click to Filter Data** to display the **Filter Parameters** dialogue as shown below


Filter Parameters
Start Time - 26/07/2012 19:56:50
End Time- 26/07/2012 22:18:53
Error
Warning
Info
Commit
Cancel

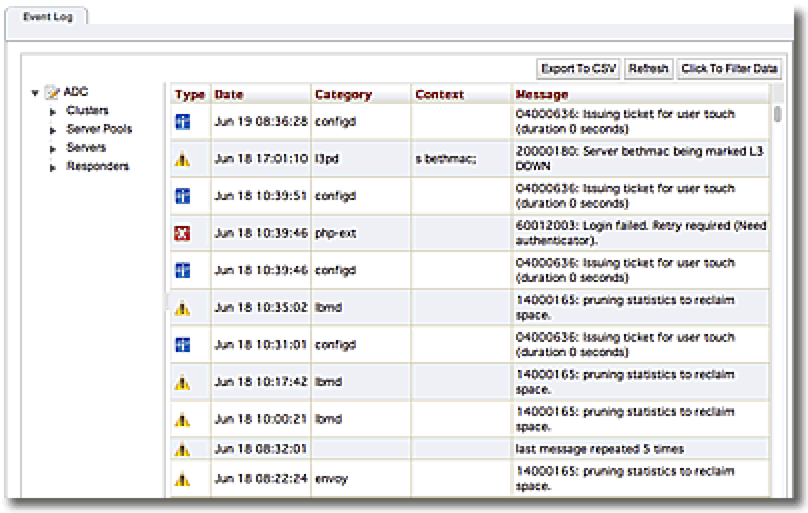
Use the sliders to specify **Start Time** and **End Time** to display events within a time frame on the Events log table.

The **Error**, **Warning** and **Info** flags can be selected to display those selected events within the time frame selected with the silders . After selecting these display options click on **Commit** to redisplay the even log.

Event Log

The event log displays events for each element configured on the Equalizer. This includes Clusters, Server Pools, Servers and Responders.

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Log and Reports** configuration tab on the left navigational pane if it is not already selected.
3. Click on the arrow (▶) beside **Logging** to expand the branch.
4. Click on **Event Log** to display the log. An example of a display is shown below.



Type	Date	Category	Context	Message
ℹ	Jun 19 08:36:28	configd		04000636: Issuing ticket for user touch (duration 0 seconds)
⚠	Jun 18 17:01:10	l3pd	s bethmac;	20000180: Server bethmac being marked L3 DOWN
ℹ	Jun 18 10:39:51	configd		04000636: Issuing ticket for user touch (duration 0 seconds)
✖	Jun 18 10:39:46	php-ext		60012003: Login failed. Retry required (Need authenticator).
ℹ	Jun 18 10:39:46	configd		04000636: Issuing ticket for user touch (duration 0 seconds)
⚠	Jun 18 10:35:02	lbmd		14000165: pruning statistics to reclaim space.
ℹ	Jun 18 10:31:01	configd		04000636: Issuing ticket for user touch (duration 0 seconds)
⚠	Jun 18 10:17:42	lbmd		14000165: pruning statistics to reclaim space.
⚠	Jun 18 10:00:21	lbmd		14000165: pruning statistics to reclaim space.
⚠	Jun 18 08:32:01			last message repeated 5 times
⚠	Jun 18 08:22:24	envoy		14000165: pruning statistics to reclaim space.

If you clicking on each individual **Clusters**, **Server Pools**, **Servers** or **Responders** the objects to the left of events table will display events for the object selected.

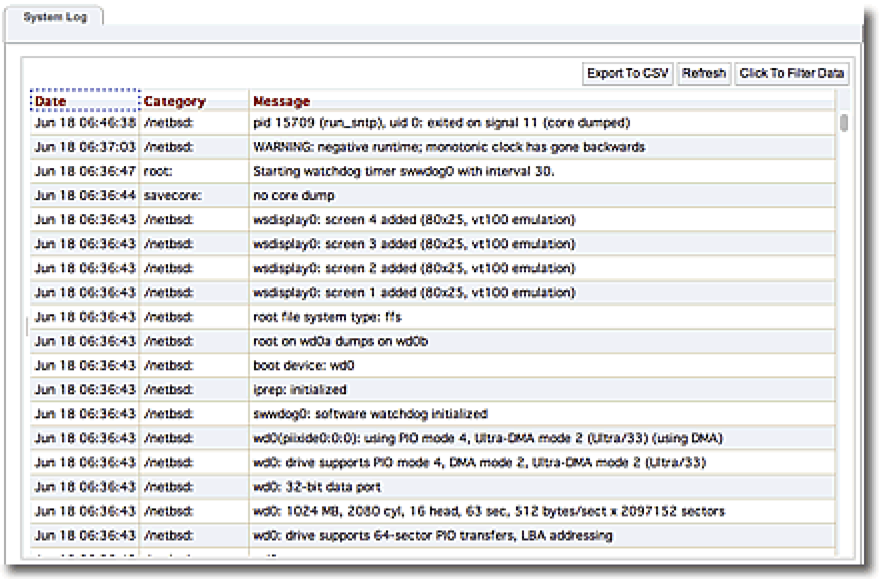
The log can be sorted by **Type**, **Date**, **Category**, **Context** and **Message** by clicking on the column heading on your browser.

Refer to "Displaying Equalizer Logs" on page 670 for instructions on displaying the **Events log** using the CLI.

System Log

Clicking on the **System Log** icon on the left will display the contents of the system log file, in which information, warnings, and error messages that are contained in the file are displayed.

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Log and Reports** configuration tab on the left navigational pane if it is not already selected.
3. Click on the arrow (▶) beside **Logging** to expand the branch.
4. Click on **System Log** to display the log. An example of a display is shown below.



Date	Category	Message
Jun 18 06:46:38	/netbsd:	pid 15709 (run_snip), uid 0: exited on signal 11 (core dumped)
Jun 18 06:37:03	/netbsd:	WARNING: negative runtime; monotonic clock has gone backwards
Jun 18 06:36:47	root:	Starting watchdog timer swwdog0 with interval 30.
Jun 18 06:36:44	savecore:	no core dump
Jun 18 06:36:43	/netbsd:	wsdisplay0: screen 4 added (80x25, vt100 emulation)
Jun 18 06:36:43	/netbsd:	wsdisplay0: screen 3 added (80x25, vt100 emulation)
Jun 18 06:36:43	/netbsd:	wsdisplay0: screen 2 added (80x25, vt100 emulation)
Jun 18 06:36:43	/netbsd:	wsdisplay0: screen 1 added (80x25, vt100 emulation)
Jun 18 06:36:43	/netbsd:	root file system type: ffs
Jun 18 06:36:43	/netbsd:	root on wd0a dumps on wd0b
Jun 18 06:36:43	/netbsd:	boot device: wd0
Jun 18 06:36:43	/netbsd:	iprep: initialized
Jun 18 06:36:43	/netbsd:	swwdog0: software watchdog initialized
Jun 18 06:36:43	/netbsd:	wd0(piixide0:0:0): using PIO mode 4, Ultra-DMA mode 2 (Ultra/33) (using DMA)
Jun 18 06:36:43	/netbsd:	wd0: drive supports PIO mode 4, DMA mode 2, Ultra-DMA mode 2 (Ultra/33)
Jun 18 06:36:43	/netbsd:	wd0: 32-bit data port
Jun 18 06:36:43	/netbsd:	wd0: 1024 MB, 2080 cyl, 16 head, 63 sec, 512 bytes/sect x 2097152 sectors
Jun 18 06:36:43	/netbsd:	wd0: drive supports 64-sector PIO transfers, LBA addressing

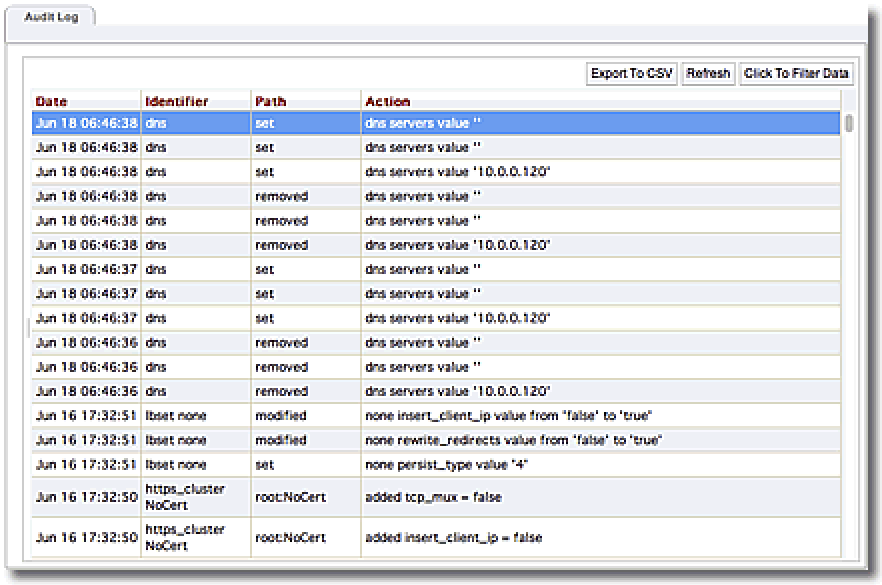
The log can be sorted by **Date**, **Category**, and **Message** by clicking on the column heading on your browser.

Refer to "Displaying Equalizer Logs" on page 670 for instructions on displaying the **System log** using the CLI.

Audit Log

Clicking on the **Audit Log** icon on the left will display the contents of the audit log showing all user activity performed on the appliance.

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Log and Reports** configuration tab on the left navigational pane if it is not already selected.
3. Click on the arrow (▶) beside **Logging** to expand the branch.
4. Click on **Audit Log** to display the log. An example of a display is shown below.



Date	Identifier	Path	Action
Jun 18 06:46:38	dns	set	dns servers value ''
Jun 18 06:46:38	dns	set	dns servers value ''
Jun 18 06:46:38	dns	set	dns servers value '10.0.0.120'
Jun 18 06:46:38	dns	removed	dns servers value ''
Jun 18 06:46:38	dns	removed	dns servers value ''
Jun 18 06:46:38	dns	removed	dns servers value '10.0.0.120'
Jun 18 06:46:37	dns	set	dns servers value ''
Jun 18 06:46:37	dns	set	dns servers value ''
Jun 18 06:46:37	dns	set	dns servers value '10.0.0.120'
Jun 18 06:46:36	dns	removed	dns servers value ''
Jun 18 06:46:36	dns	removed	dns servers value ''
Jun 18 06:46:36	dns	removed	dns servers value '10.0.0.120'
Jun 16 17:32:51	lbset none	modified	none insert_client_ip value from 'false' to 'true'
Jun 16 17:32:51	lbset none	modified	none rewrite_redirects value from 'false' to 'true'
Jun 16 17:32:51	lbset none	set	none persist_type value '4'
Jun 16 17:32:50	https_cluster NoCert	root:NoCert	added tcp_mux = false
Jun 16 17:32:50	https_cluster NoCert	root:NoCert	added insert_client_ip = false

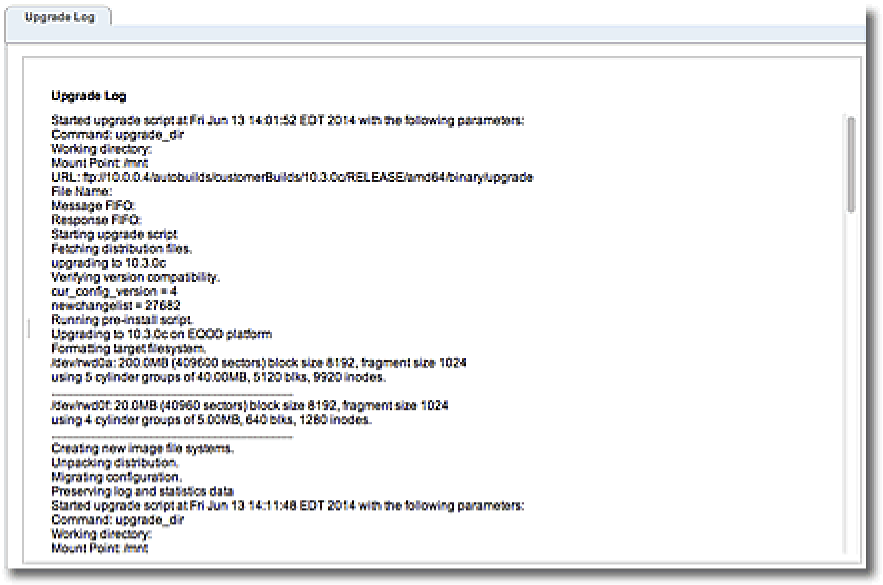
The log can be sorted by **Date**, **Identifier**, **Path**, and **Action**.by clicking on the column heading on your browser.

Refer to "Displaying Equalizer Logs" on page 670 for instructions on displaying the **Audit log** using the CLI.

Upgrade Log

Clicking on the **Upgrade Log** icon on the left will display the contents of the upgrade log with upgrade details of previous software upgrades on your appliance.

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Log and Reports** configuration tab on the left navigational pane if it is not already selected.
3. Click on the arrow (▶) beside **Logging** to expand the branch.
4. Click on **Upgrade Log** to display the log. An example of a display is shown below.



Upgrade Log

```
Started upgrade script at Fri Jun 13 14:01:52 EDT 2014 with the following parameters:
Command: upgrade_dir
Working directory:
Mount Point: /mnt
URL: ftp://10.0.0.4/autobuilds/customerBuilds/10.3.0c/RELEASE/amd64/binary/upgrade
File Name:
Message FIFO:
Response FIFO:
Starting upgrade script
Fetching distribution files.
upgrading to 10.3.0c
Verifying version compatibility.
cur_config_version = 4
newchangelist = 27682
Running pre-install script.
Upgrading to 10.3.0c on EQOD platform
Formatting target filesystem.
/dev/rwd0a: 200.0MB (409600 sectors) block size 8192, fragment size 1024
using 5 cylinder groups of 40.00MB, 5120 blks, 9920 inodes.
--------------------------------------------------------
/dev/rwd0f: 20.0MB (40960 sectors) block size 8192, fragment size 1024
using 4 cylinder groups of 5.00MB, 640 blks, 1280 inodes.
--------------------------------------------------------
Creating new image file systems.
Unpacking distribution.
Migrating configuration.
Preserving log and statistics data
Started upgrade script at Fri Jun 13 14:11:48 EDT 2014 with the following parameters:
Command: upgrade_dir
Working directory:
Mount Point: /mnt
```

Remote System Logging

Remote system logging is enabled using commands in the global CLI context using the **syslog-server** and **syslog** commands, and in the GUI from the **Log and Reports** configuration tab.

Enabling Remote System Logging (GUI)

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Log and Reports** configuration tab on the left navigational pane if it is not already selected.
3. Click on the arrow (▶) beside **Logging** to expand the branch.
4. Select **Remote Syslog** to display the Remote Syslog entry form on the right.
5. Type the IP address or name of the remote logging server into the **Syslog Server** text box. If one is already supplied, skip this step.
6. Enable the **Syslog Enable** option.
7. Click on **Commit**.

Disabling Remote System Logging (GUI)

1. Verify that you are logged into the GUI. If not, log in as described in "Logging In" on page 238.
2. Select the **Log and Reports** configuration tab on the left navigational pane if it is not already selected.
3. Click on the arrow (▶) beside **Logging** to expand the branch.
4. Select **Remote Syslog** to display the Remote Syslog entry form on the right.
5. Do one or both of the following:
    a. Remove the contents of the **Syslog Server** text box.
    b. Turn off the **Syslog Enable** check box.
6. Click **Commit**.

Enabling Remote System Logging (CLI)

When setting up the system for the first time, remote logging is enabled by a single command:

```
eqcli > syslog-server IPaddr_or_name
```

Substitute the IP address or hostname of a working **syslog()** server for ***IPaddr_or_name***.

If the remote syslog server is later removed using the **no** form of the **syslog-server** command, use the syntax shown above to re-enable remote logging.

If remote logging is later disabled using the **syslog disable** command, use **syslog enable** to re-enable it.

Disabling Remote System Logging (CLI)

To disable remote logging without removing the IP address or name of the current remote logging server, enter:

```
eqcli > syslog disable
```

Alternatively, removing the IP address or name of the current remote logging server will also automatically disable remote logging:

```
eqcli > no syslog-server
```

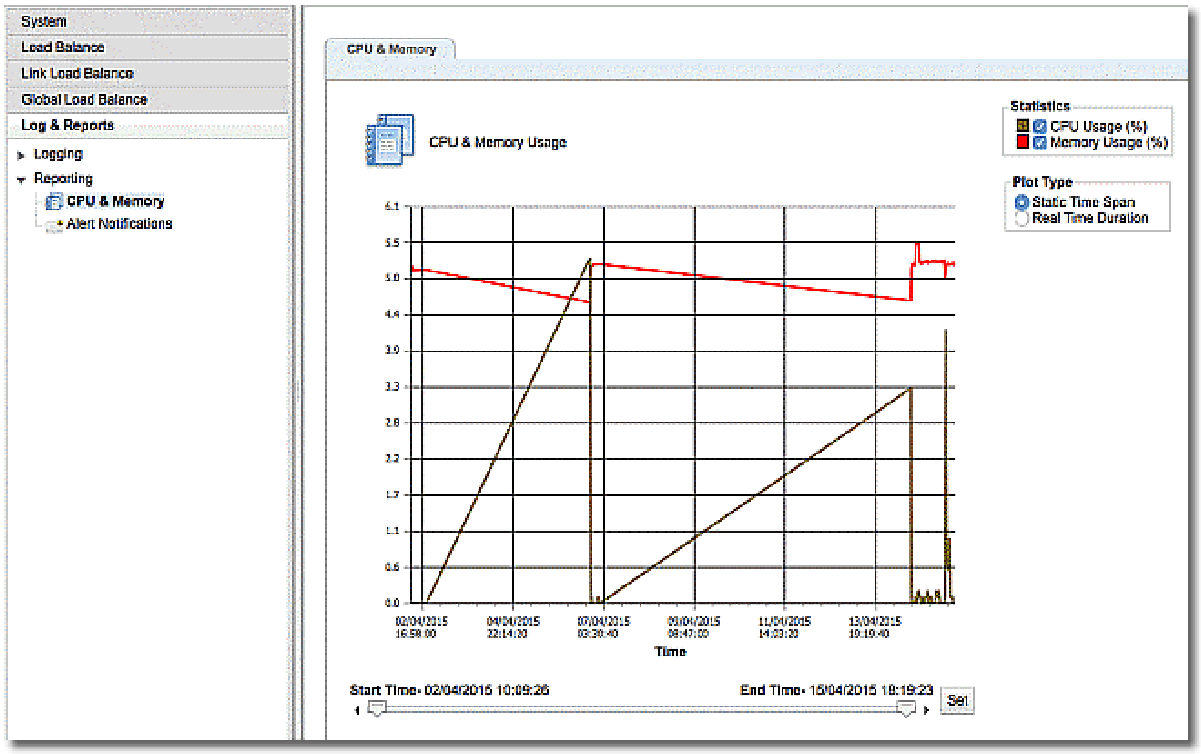
Usage Reporting and Alert Notifications

CPU and Memory Usage Reporting

The CPU & Memory Usage display indicates:

- **CPU Consumption** - The average percent of non-idle CPU time over the selected time period.
- **Memory Utilization** - The average percent of in-use memory over the selected time period.

1. Log in to the GUI.
2. Select **Log and Reports > Reporting > CPU & Memory** on the left navigational pane. The CPU & Memory Usage screen shown below will be displayed.



Statistics: ☑ CPU Usage (%) ☑ Memory Usage (%)

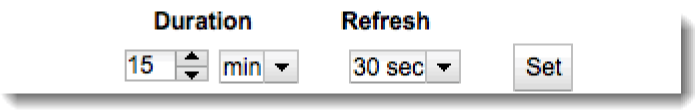
Plot Type: [● Static Time Span] ○ Real Time Duration

Start Time- 02/04/2015 10:09:26

End Time- 15/04/2015 18:19:23

The default plot displays a **Static Time Span** plot of current CPU and Memory usage values from the beginning of the available data to the latest sample. This can be adjusted by using the slider to adjust **Start Time** and **End Time** and clicking on **Set**.

When you select **Real Time Duration**, the slider changes to selectors where you can configure the **Duration** of the plotted data and **Refresh** time. Click on **Set** when you make a selection.



Duration: 15 min

Refresh: 30 sec

Displaying Alert Notifications

Notification IDs are assigned a monotonically increasing number starting with 1, up to a maximum of 200. After 200, the notification ID wraps back to 1. After wrapping all previously uncleared notifications are overwritten as the ID is re-used.

Displaying all pending alert notifications in the CLI

Enter the following:

```
eqcli > show notification
```

The notifications are listed in the order in which the alerts were generated (not configured).

```
eqcli > show notification
ID  Time Stamp       Type          Obj Type  Obj Name    Alert Name
1   Apr 15 11:48:22  state_change  fogrp     Unassigned  al_allfogrps
2   Apr 15 11:48:45  state_change  server    testing123  al_allservers
3   Apr 15 11:48:46  state_change  si        srv1        al_allsis
4   Apr 15 11:48:47  state_change  server    srv1        al_allservers
5   Apr 15 11:52:40  state_change  fogrp     Unassigned  al_allfogrps
eqcli >
```

To display the *first* notification enter:

```
eqcli > show notification first
```

The following is an example of the first alert above for the state change for **swport01**.

```
eqcli > show notification first
Notification ID       : 1
Generation Timestamp  : Apr 15 11:48:22
Alert Type            : state_change
Alert Subtype         : Standalone
Alert Name            : al_allfogrps
Object Type           : fogrp
Object Name           : Unassigned
Message               : 47000194: Local Peer Tatooine has entered
Standalone-!Owner mode
eqcli >
```

To show the first notification matching one or more filters enter:

```
eqcli > show notification first alert_type alerttype object_type objecttype object_name
objectname
```

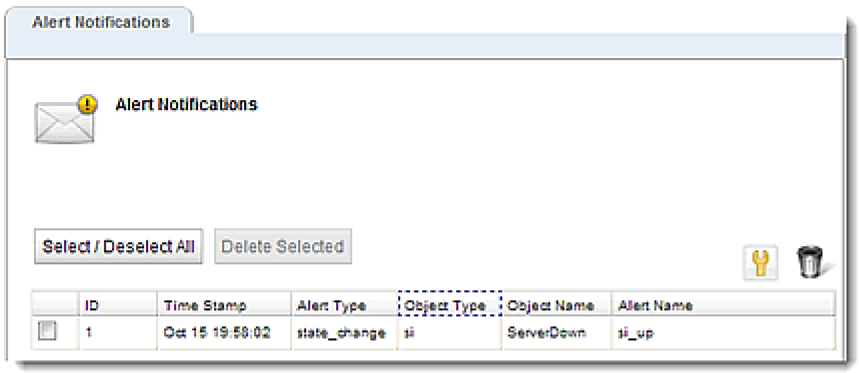
If the **object_name** is specified, **object_type** must also be specified.

The following is an example showing the state change for **swport01** above.

```
eqcli > show notification first alert_type state_change object_type
interface object_name swport01

Notification ID  : 1
Alert Type       : state_change
Alert Subtype    : Up
Alert Name       : al_switch
Object Type      : interface
Object Name      : swport01
Message          : 50000197: Port 1 has become ACTIVE
eqcli >
```

Displaying all pending alert notifications in the GUI

1. Click on the **Logs and Reports**configuration tab in the left navigational pane.
2. Click on the arrow (▶) next to **Reporting** to expand the branch.
3. Click on **Notification**. All configured **Alert Notifications** will be displayed on the right. In addition, the number of **pending notification alerts** will be displayed on the beneath the **Log Out** and **Help** buttons on the upper right of the GUI.



	ID	Time Stamp	Alert Type	Object Type	Object Name	Alert Name
☐	1	Oct 15 19:58:02	state_change	si	ServerDown	si_up

Setting the Interval

The alert interval is set using the CLI. To set the interval at which alert notifications are displayed on the CLI enter the following:

```
eqcli> user-tou*> alert_interval integer
```

By default **alert_interval** is 1 minute. The maximum is 86400 (1 day).

Removing all notification alerts in the CLI

```
eqcli > no notification all
```

Removing individual notifications using the ID number by entering:

```
eqcli > no notification id-number
```

Removing notification alerts in the GUI

1. Select the **System** configuration tab on the left navigational pane.
2. Click on the arrow (▶) next to **Alerts** to expand the branch.
3. Click on **Notification**. All configured **Alert Notifications** will be displayed on the right.
4. Select an alert and click on **Delete Selected** or select it and click on . 🗑.

Chapter 21
Failover

Sections within this chapter include:

Understanding Failover

Failover allows a second ADC to take over quickly if the primary unit fails allowing data and applications to be continuously delivered without interruption. The load balancing pair can be configured into an active/standby (also known as Active/Passive) mode or in Active/Active mode.

In an Active/Passive failover configuration, two Equalizers are configured into active and passive roles, with the active unit serving cluster traffic. A "failover" is said to occur when the active Equalizer stops processing client requests and the passive Equalizer starts processing cluster traffic.

When two Equalizers are configured into this type failover configuration, they form what is called a "failover pair". An Equalizer in a failover pair is called a "peer". At any given time, only one of the Equalizer s in a failover pair actually services requests sent to the cluster IP addresses defined in the configuration -- this unit is called the "active peer" or the "current primary" Equalizer in the pairing. The other Equalizer, called the “passive peer” or "current backup", does not process any client requests.

Both units continually send "heartbeat probes" or "failover probes" to one another. If the current primary does not respond to heartbeat probes, a failover occurs. In this scenario the current backup Equalizer assumes the primary role by assigning the cluster IP addresses to its network interfaces and begins processing cluster traffic.

In an Active/Active failover configuration, clusters are active on both peers in the configuration. Active/Active failover situations are the same failure situations that cause a peer to take over all the cluster and floating IP addresses in an Active/Passive failover configuration. A healthy peer will take over all of the cluster and failover IPs. This type of failover configuration is only available when using two Equalizers using EQ/OS 10.

How Equalizer Determines if it Should Assume the Primary Role

Equalizer expects to see a heartbeat from a failover peer (all failover peers in Active/Active Failover configuration [See Configuring Active/Active Failover Between Two Systems]).within a "heartbeat interval" (time in seconds) on every subnet where a heartbeat flag is configured. When a "Failed Probe Count" value is reached – that is, when that number of heartbeat probe intervals has elapsed without receiving a heartbeat from a peer -the backup unit will attempt to assume primary role.

Equalizer will:

1. Check to see if any of the cluster or failover IP addresses defined in its configuration are already on the network -using ICMP requests/replies.
2. Perform checks on its locally defined networks and determine the number of subnets on which it has connectivity. It compares this information to the connectivity information in the last heartbeat received from the peer that reached the "Failed Probe Count" value.
    a. If the Global "Failed Probe Count" on the failover configuration = 0, then the "Failed Probe Count" configured on the subnet will be used to determine when failover occurs.
    b. If the Global "Failed Probe Count" is reached BEFORE the "Failed Probe Count" configured on the subnet, then failover will occur.
    c. If the "Failed Probe Count" configured on the subnet is reached BEFORE the Global "Failed Probe Count" a failover will occur.
3. If the load balancer determines that no other systems own any of its cluster or failover IP addresses AND it has better connectivity than the peer, it becomes "Primary" otherwise, it will remain a "Backup".

Heartbeating MUST be established on an interface before failover can occur. If failover is configured between two peers, and the cable is removed from the primary peer, the backup will assume the role as the new primary. If the cable is removed from the "new" primary BEFORE heartbeating has been reestablished with the previous primary, this peer WILL NOT failover since heartbeating was never reestablished.

Releases Supported for Failover with EQ/OS 10

An Equalizer running EQ/OS 10 can be configured into failover with another Equalizer running either of these releases:

- EQ/OS 10
- EQ/OS 8.6 (latest release)

Note - Failover is *not* supported between EQ/OS 10 and any release prior to EQ/OS 8.6.0c.

Failover Between Two EQ/OS 10 Systems

The procedures in this section show you how to configure failover using two Equalizers running EQ/OS 10.

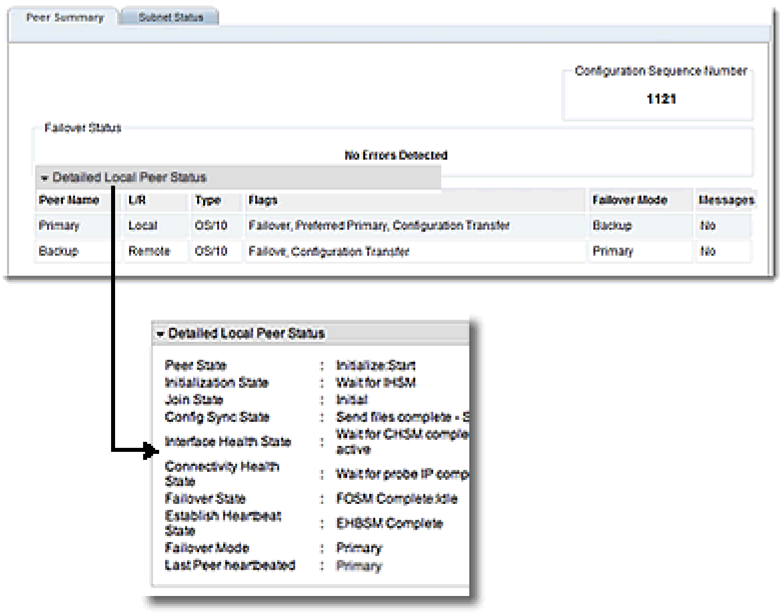
Displaying Failover Configuration

You can display a summary of the current failover configuration on your local Equalizer in both the CLI and the GUI.

In the CLI enter `eqcli >` **`show failover`**. The following is an example of the failover display:

```
eqcli > show failover
Local Peer Failover Information
Command subnet:          Vlan v2, Subnet sn172
Failover:                Enabled
Mode:                    Active/Passive
Preferred Primary:       Yes
Config Sync:             Enabled
12000003: You have 1 pending alert notification.
eqcli >
```

In the GUI select **System > Failover** on the left navigational pane. The following is an example of the Peer Summary display.



Configuration Sequence Number

1121

Failover Status

No Errors Detected

Detailed Local Peer Status

Peer Name	L/R	Type	Flags	Failover Mode	Messages
Primary	Local	OS/10	Failover, Preferred Primary, Configuration Transfer	Backup	No
Backup	Remote	OS/10	Failove, Configuration Transfer	Primary	No

Detailed Local Peer Status

Peer State	:	Initialize:Start
Initialization State	:	Wait for IHSM
Join State	:	Initial
Config Sync State	:	Send files complete - S
Interface Health State	:	Wait for CHSM comple active
Connectivity Health State	:	Wait for probe IP comp
Failover State	:	FOSM Complete:Idle
Establish Heartbeat State	:	EHBSM Complete
Failover Mode	:	Primary
Last Peer heartbeated	:	Primary

Types of Failover

Active/Passive Failover

When two Equalizers are configured into Active/Passive failover, they form a "failover pair". An Equalizer in a failover pair is called a "peer". At any given time, only one of the load balancers in a failover pair is actually servicing requests sent to the cluster IP addresses defined in the configuration -- this unit is called the "active peer" or the "current primary" load balancer in the failover pair. The other load balancer, called the "passive peer" or "current backup", does not process any client requests.

Both units continually send "heartbeat probes" or "failover probes" to one another. If the current primary does not respond to heartbeat probes, a failover occurs. In this scenario the current backup load balancer assumes the primary role by assigning the cluster IP addresses to its network interfaces and begins processing cluster traffic.

Refer to Configuring Active/Passive Failover (CLI) or Configuring Active/Passive Failover (GUI) for procedures on configuring Active/Passive failover.

Active/Active Failover

Active/Active (A/A) failover is a feature of EQ/OS 10 that allows clusters to be Active on *both* peers that are in failover. For the same failure situations that cause a peer to take over all the cluster and floating IP addresses in an Active/Passive failover configuration, A/A operates the same way - that is that a healthy peer takes over all of the cluster and failover IPs. However, if and when the "sick" peer is healed, there is no automatic return migration of the clusters and the user needs to invoke a "rebalance" command to cause this to happen.

Active/Active failover can be configured between two EQ/OS 10 Systems only.

Refer to Configuring Active/Passive Failover (CLI) or Configuring Active/Passive Failover (GUI) for procedures on configuring Active/Active failover.

N+1 Failover

N+1 Failover is a feature of EQ/OS 10 where the failover configuration consists of multiple active peers ("N") plus 1 passive peer. In this type of failover configuration, the Equalizer clusters are instantiated on all "N" peers and organized into failover groups. If the passive, or backup peer's connectivity for a failover group's resources is judged to be "healthier" that the peer on which the group is running, then the group fails over to the passive peer, which becomes the Primary peer.

N+1 failover can be configured between two EQ/OS 10 Systems only.

Peer Failover Modes

The Failover Mode is the current failover condition of an Equalizer peer or Failover Group when configured into Active/Passive, Active/Active, or N+1 failover configuration. It is shown both in the an eqcli display and on the GUI.

The following table lists the possible Failover Modes that can be displayed along with a description of each.

Failover Mode	Description
Standalone-Owner	Indicates that the system is in Standalone mode and owns any cluster and/or failover IPs.
Standalone-!Owner	Indicates the system is in Standalone mode and does not own any cluster and/or failover IPs.
Isolated	The system is configured in failover, however, there is no connectivity to it.
Primary	The system is the Primary for all Failover Groups. **Note**: In Active/Passive failover, there is only 1 Failover Group.
Backup	The system is the Backup for all Failover Groups. **Note:** In Active/Passive failover, there is only 1 Failover Group.
Mixed	The system is in Active/Active failover and is the Primary for at least one Failover Group and Backup for at least 1 Failover Group.
Initializing	The system is still initializing and has not yet settled into any of the above states.

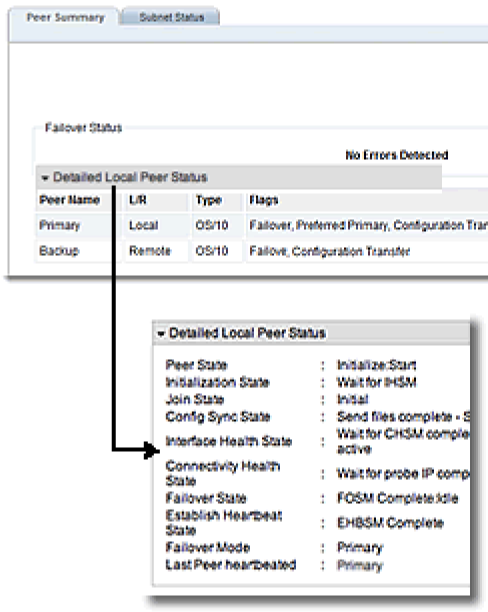
In the CLI, the failover mode is displayed in the **F/O Mode** column. To display the Failover Mode in the CLI, enter the following:

```
eqcli > show fogrp

F/O Group Name         F/O Group ID    F/O Mode        Primary Peer
Unassigned             0               Not Used
fo-group1              1               Primary         Eq-A
fo-group2              2               Backup          Eq-B
```

If you are using Active/Active or N+1 failover, the following is an example of where the F/O Mode is displayed for the failover groups as follows:

```
eqcli > show fogrp

F/O Group Name         F/O Group ID    F/O Mode        Primary Peer
Unassigned             0               Not Used
fo-group1              1               Standalone
fo-group2              2               Standalone
```

To display the **Failover Mode** in the **Failover Status** screen in the GUI, click on the **System** configuration tab in the left navigational pane if it has not already been selected. Click on **Failover** to display the **Peer Summary** screen. An example is shown below:



Configuration Sequence Number

1121

Failover Status

No Errors Detected

Detailed Local Peer Status

Peer Name	L/R	Type	Flags	Failover Mode	Messages
Primary	Local	OS/10	Failover, Preferred Primary, Configuration Transfer	Backup	No
Backup	Remote	OS/10	Failove, Configuration Transfer	Primary	No

Detailed Local Peer Status

Peer State	:	Initialize:Start
Initialization State	:	Wait for IHSM
Join State	:	Initial
Config Sync State	:	Send files complete - S
Interface Health State	:	Wait for CHSM comple active
Connectivity Health State	:	Wait for probe IP comp
Failover State	:	FOSM Complete:Idle
Establish Heartbeat State	:	EHBSM Complete
Failover Mode	:	Primary
Last Peer heartbeated	:	Primary

Failover Constraints

Before you begin configuring failover, you must do the following:

1. Ensure that the VLAN configuration on both EQ/OS 10 Equalizer is *exactly* the same.
2. In some cases there may appear to be an issue where the Primary and Backup Equalizers are in a conflict over Primary. Any switch, such as one from Cisco or Dell, that comes with Spanning Tree Protocol enabled by default can cause a communication problem in a failover configuration. This problem occurs at boot up because the switch disables its ports for roughly 30 seconds to listen to BPDU (Bridge Protocol Data Unit) traffic. The 30 second pause causes both Equalizers to attempt to become the primary unit, and the default backup continually reboots. To repair this condition, either disable Spanning Tree Protocol or enable PortFast for the ports connected with the Equalizers. This enables the ports to act as normal hubs and accept all traffic immediately.
3. When configuring VLAN subnets, the following must be true:
   - the **heartbeat** flag must be enabled on *at least one* VLAN
   - the **command** flag must be enabled on *exactly one* VLAN
   - the **Failover IP** (**virt_addr** in the CLI) parameter must be set on all VLAN subnets that have the **heartbeat** or **command** flags enabled

4. Other important notes:
    - Run http on the failover IP address, not the VLAN IP address.
    - Only make changes when logging in over the failover IP address.
    - If you run GUI/SSH on the VLAN IP addresses on both peers, then do NOT go back and forth between peers making configuration changes, unless you verify that each change is transferred before you making a change on the “other” unit.
5. A note about GX and LX series Equalizers configured into failover.
    - The VLAN and Link Aggregation configurations are not synched. Furthermore, the VLAN ports are not compared, I.e., the ports on the VLANs on each system in failover do not have to be the same. Since aggregation is just another flavor of ports, the aggregation configurations do not have to be the same, either.

Configuration Synchronization Constraints

Whenever a configuration change is made on either EQ/OS 10 failover unit, the failover subsystem synchronizes the configuration by transferring the configuration file to the other unit over the VLAN subnet that has the **command** flag enabled.

If the command flag (Command Transfer in the GUI) is NOT set for any VLAN, the system will use the first VLAN in the configuration file for Configuration transfer.

If you are currently using EQ/OS 10.3.2d , EQ/OS version 10.2.x or higher must be installed on Equalizers used as failover peers for configuration synchronization to occur correctly. It is highly recommended that you upgrade the OS version of all of the failover peers to 10.2.x or higher.

Failover Configuration Transfer

The following applies when changing the default value of the **Configuration Transfer** peer option (**fo_config_xfer** in the CLI):

1. By default, this option is **enabled** on all peer definitions and should usually be left enabled unless there is a specific reason that the configuration of the two failover units must be different. [Note that it is no longer necessary, as it was in EQ/OS Version 8, to disable configuration file transfer during upgrades. It is also not necessary when operating in failover with a Version 8 Equalizer.]
2. If this option is **disabled** in the local peer definition, then configuration file transfer will not be initiated or accepted by that system.
3. When **Configuration Transfer** is disabled between two peers and a VLAN change is made on either or both systems, then failover between the units will be disabled because of a VLAN mismatch. There will be errors evident in the GUI (on the **Peer Summary** screen) and in the CLI (`eqcli >` **`show peer`** ***`peer_name`*** output). To re-enable failover, do the following:
    a. Ensure that the VLAN/subnet configuration is the same on both units (with the exception of names, VLAN IP addresses, and assigned ports).
    b. Enable configuration file transfer between the two peers by enabling the **Configuration Transfer** (**fo_config_xfer** in the CLI) option in the local peer definition on both peers.
    c. If the sequence number of the configuration file is the same on both units, then you must also make a configuration change on one of the units so that the configuration file with the highest number is transferred to the other unit. Once both peers determine that they have the same VLAN configuration by comparing the newly transferred configuration file, failover will be re-enabled.
    d. Ensure that the settings of a remote peer's flags are synchronized with the remote peer when configuration transfer occurs.

Synchronization Notes

1. Failover does not require the same set of VLANs on all Peers. Therefore, a failover group associated with a VLAN existing on one Peer cannot be configured into failover with other peers that do not have the same VLAN configured. All instances of a VLAN mismatch will be logged.
2. A failover group associated with a subnet existing on one peer cannot be configured into failover with other peers that do not have the same subnet configured. All instances of a subnet mismatch will be logged.
3. The following subnet parameters must be the same for each peer. They will be synchronized amongst Peers just as the non-network parameters are:
   - heartbeat interval
   - failed heartbeat (strike) count
   - floating IP address
4. The following subnet parameters and flags do not need to be the same for each peer. This will not affect failover operation, however, they will be checked and a warning message will be logged.
   - services
   - outbound NAT
   - heartbeat
   - command
5. The following VLAN parameters need not be the same for each peer. They will not be checked and therefore no error is logged if they do not match:
   - MTU
   - interface instances
   - aggregated interfaces
6. The following peer parameters must be the same on all peers configured into failover. However, they will be synchronized amongst the Peers just as the non-network parameters are:
   - receive timeout
   - connect timeout
   - heartbeat interval
   - retry interval
   - strike count
7. SNI objects created for HTTPS clusters are part of failover \synchronization. SNI objects ("Server Name Indication (SNI)" on page 402) have a name; they contain a server name in FQDN syntax and a pointer to a certificate (the name of the certificate in the global certificate store).
8. Restore and Failover synchronization:
   - Assume two units (A, B) are in failover.
   - Unit A is taken offline.
   - A new unit (C) is brought online to replace unit A, and a backup archive from unit A is restored onto unit C.
   - Unit C is then powered off, and unit A is brought back online. Iif unit A was only disconnected from the network and not shut down when it was taken off line, a reboot of unit A is required in order to synchronize failover between the two units properly.

Server / Gateway Availability Constraint

For failover to initialize correctly, at least one server or gateway configured on a subnet defined on Equalizer must be responding to ARP (Address Resolution Protocol) requests. Otherwise, Equalizer will remain in the "initializing" failover state and will not assume the backup or primary role.

You can check server availability status in eqcli using this command:

```
eqcli > server server_name stats
```

Server availability can also be checked in the GUI by looking at the icons next to the server name in the left pane object tree. Unavailable servers have an exclamation point icon displayed to the right of the server name.

Failover Peer Probes and Timeouts

When Equalizers are configured into a failover group, they continually probe (or heartbeat) each other so that a backup peer can assume the primary role, should the active primary unit become unreachable.

Heartbeat probes are performed over a long-lived TCP connection. Whenever Equalizer starts heartbeating a peer, it opens a heartbeat connection to the peer which remains open for as long as the two systems are operational and have network connectivity. All heartbeats between the two peers will occur over this long-lived connection.

Once failover is configured, it is the system with the "greater" system ID that always starts the heartbeating process. For example, of one "sysid" is "003048BC2C8A" and the other is "003048D52AA2". The second "sysid" has a higher hex value and will start the heartbeating process.

Failover Timeout Parameters

Heartbeat timeout probes are configured using the following parameters and are defined at the peer levels.

To access these timeouts using the GUI, log in and select the **System** configuration tab the then select the arrow (▶) next to **Failover** to expand the branch. Click on **Peers**to display the **Failover > Parameters** screen. Use the sliders to adjust timeouts.

To configure timeouts using the CLI, log in and access the peer context. Configure the timeouts using the guidelines and parameters descriptions below and also in "Peer Commands" on page 201.

GUI Parameter (CLI Parameter)	Description
Heartbeat Interval (hb_interval)	The time in seconds (default:2) between sending and receiving heartbeat probes between Equalizer and a peer. Each peer expects to receive a heartbeat probe from other peers that have a failover IP address on a subnet within this interval.
Failed Probe Count (strike_count)	The number of successive failed heartbeats that must occur before a peer is marked "down" (default:3). A heartbeat is considered to have failed whenever the **Heartbeat Interval** has elapsed and no probe has been received from a peer during that interval. If the **Failed Probe Count** reaches its specified maximum value on the subnet, or if the **Failed Probe Count** reaches 1 on all subnets in a multi-subnet network configuration, then a failover can occur. If the subnet **Failed Probe** count is 0, then the global **Failed Probe Count (strike_count)** is used
Retry Interval (retry_interval)	Time in seconds (default:5) between checks for changes in Equalizer's configuration, for the purpose of determining whether a configuration transfer to the remote peer is required. The **Retry Interval** is also used when any failover operation other than heartbeating fails. For example, if the system is rebooted and a VLAN health check fails, we try again

GUI Parameter (CLI Parameter)	Description
Connectivity Retry Count (`conn_retry_count`)	This controls the number of times that a peer will try to re-gain connectivity with a remote peer before becoming the primary peer for any failover groups running on the remote peer. (Default: 8) If spurious failovers occur, this parameter can be adjusted, for example, to allow for more retries before a failover occurs.
Connect Timeout (`conn_timeout`)	Time in seconds (default:1) to wait for a TCP connection to be established between peers.
Receive Timeout (`recv_timeout`)	Time in seconds (default:1) after a TCP connection is established, of how long a peer waits for the other peer's response.

Failover Occurs If...

- The number of failed probes on any single subnet equals or exceeds the **Failed Probe Count** for that subnet.

OR

- The number of subnets with a **Failed Probe Count** greater than 0 equals or exceeds the global **Failed Probe Count.**

Modifying Failover Timeouts in Production

When an failover pair is actively serving traffic, any changes to the global or subnet failover parameters could result in a failover if you do not perform them in the correct manner.

Note - These parameters are not currently synchronized.

To prevent a failover from occurring , proceed with the following to prevent an inadvertent failover:

1. Disable failover on both units by disabling the failover flag on the local peers on both units; first on the current primary unit and then the current backup. Refer to "Configuring Active/Passive Failover (CLI)" on page 714 or "Configuring Active/Passive Failover (GUI)" on page 724 for details.
2. Make the changes to failover timeouts required on both systems, in any order. Refer to "Failover Peer Probes and Timeouts" on page 695 if necessary.
3. Re-enable failover by checking the failover flags on the on the primary Equalizer's local peer and then on the backup.

Peer, Interface, Subnet States and Substates

The following table lists the various valid states/substates for Peers, Interfaces, and Subnets.

Peer State	**Peer Substate**	**Explanation**
Send Join	Start	Communicating with remote Peer to join into a Failover configuration.
	Failed	Attempt to configure remote Peer into a failover configuration failed.
Configure	Success	The local and remote Peers have equivalent network configurations.
	Failed	The local and remote Peers do not have equivalent network configurations.
Initialize	Success	All interfaces have connectivity with the remote Peer.
	Partial	Some of the interfaces do not have connectivity with the remote Peer.
	Failed	None of the interfaces have connectivity with the remote Peer.
Configure heartbeating	Waiting	An attempt is being made to configure heartbeating on all heartbeating subnets.
	Failed	Heartbeating could not be configured on 1 or more heartbeating subnets on.
Heartbeating	Start	The local and remote Peers are heartbeating on all heartbeating subnets.
	Partial	The local and remote Peers are heartbeating on 1 or more heartbeating subnets.
	Suspended	The local and remote Peers are no longer heartbeating on any heartbeating subnets. The local Peer is attempting to re-establish heartbeating with the remote Peer.

Failover Between EQ/OS 8.6 and EQ/OS 10

The procedures in this section show you how to configure failover using an Equalizer running EQ/OS 8.6 as the preferred primary and an Equalizer running EQ/OS 10 as the preferred backup.

Guidelines for Upgrading a Failover Pair from EQ/OS 8.6 to EQ/OS 10

The preferred method of upgrading the OS from EQ/OS 8.6 to EQ/OS 10 is:

1. Upgrade the preferred backup Equalizer to EQ/OS 10, while it is in backup failover mode. Refer to "EQ/OS 8.6 Upgrade Procedure" on page 77 for instructions.
2. Configure the EQ/OS 10 unit and the EQ/OS 8.6 unit into failover, with the EQ/OS 8.6 unit as the preferred primary. Run the two units in failover to validate that the units are communicating properly as observed on the EQ/OS 10 unit.
3. Test failover to EQ/OS 10 by rebooting the EQ/OS 8.6 system so that the EQ/OS 10 system becomes the current primary. Then, reboot the EQ/OS 10 unit to test failover to the EQ/OS 8.6 unit.
4. While the EQ/OS 8.6 system is the current backup, upgrade it to EQ/OS 10. Verify that the units are communicating properly. Reboot each unit to ensure that the other unit assumes the primary role.

EQ/OS 8.6 Failover Constraints

An Equalizer running EQ/OS 10 can be configured into failover with an Equalizer running EQ/OS 8.6, subject to these constraints:

- The failover configuration is limited to two (2) VLANs only; both Equalizers must be configured with at most two VLANs.
- Configuration synchronization is *not supported* (that is, the **dont transfer** option must be enabled on both units). The configurations on both devices must be updated manually to maintain equivalent configurations, meaning the following:

The VLAN configuration on the EQ/OS 10 Equalizer must match that of the EQ/OS 8.6 Equalizer. This includes the subnets and failover IP addresses set on each system. [Note that although EQ/OS 10 supports multiple subnets per VLAN, you can only configure one subnet per VLAN when enabling failover with an EQ/OS 8.6 system.]

Clusters and servers must be configured equivalently on both systems. This means that cluster and server IP addresses, ports, and parameters must be set to the same values on both systems. The difference is that on EQ/OS 8.6, servers are assigned directly to clusters, while on EQ/OS 10, servers are assigned to server pools and then to clusters. The important point is that the server pool used in a cluster on EQ/OS 10 must be configured with the same servers used in the equivalent cluster on the EQ/OS 8.6 system.

- On the EQ/OS 8.6 system, failover must be configured manually as shown in the procedure below (i.e., you cannot use the Failover Wizard).

Server Availability Constraint

For failover to initialize correctly, at least one server or gateway configured on a subnet defined on Equalizer must be responding to ARP (Address Resolution Protocol) requests. Otherwise, Equalizer will remain in the "initializing" failover state and will not assume the backup or primary role.

On EQ/OS 10, you can check server availability status in `eqcli` using this command:

```
eqcli > server server_name stats
```

On EQ/OS 8.6 and EQ/OS 10EQ/OS 10, server availability can be checked in the GUI by looking at the icons next to the server name in the left pane object tree. In EQ/OS 8.6, unavailable servers have an exclamation point (**!**) icon displayed over the server icon. In EQ/OS 10, unavailable servers have an exclamation point icon displayed to the right of the server icon.

Enable Failover of EQ/OS 8.6 to EQ/OS 10

Note - EQ/OS 8.6 uses the web-based GUI for all failover configuration.

On EQ/OS 10, you can set up failover using either the CLI or the GUI, but failover status information is currently only visible in the CLI **peer** context.

The following procedure describes how to enable failover between one Equalizer running EQ/OS 10 and one running EQ/OS 8.6.

1. Configure the Equalizer running EQ/OS 8.6:
   a. Perform initial system configuration as outlined in the *EQ/OS 8.6 Installation and Administration Guide*.
   b. Create all required VLANs, clusters, servers, etc., required for your configuration.
   c. Ensure that the configuration is working properly. In particular, make sure that at least one server is active (that is, marked "up" in the GUI). Failover will not properly initialize if Equalizer cannot successfully probe at least one server.
2. Add the VLAN IP addresses. Once you verify that the EQ/OS 8.6 system is working properly in standalone mode, open the **Equalizer > Networking > VLAN Configuration** tab and do the following for *each* VLAN in the table:
   a. Click the modify icon .
   b. Enter a **Failover IP** address and **Failover Netmask** for the VLAN.
   c. Enable the **Use IP for Failover Heartbeat** check box.
   d. Click **Commit**.
3. Configure failover peers on the EQ/OS 8.6 system.
   a. Click **Mode: Standalone** at the top of the left frame to open the **Failover > Required** tab.
   b. Uncheck (turn off) the **Disable Failover** check box.
   c. Check (turn on) the **Preferred Primary** check box in the **This Equalizer** section.

d. Do the following in the **Peer Equalizer** section, :

- Enter any name you like for **Equalizer Name** and a string of characters for the **Signature.** The signature of the EQ/OS 10 peer can be found by:

  Entering the peer context and entering `eqcli peer-eq_*>` **`show peer`** using the CLI
  or
  On the GUI by clicking on the **System** configuration tab on the left navigational pane and expanding the **Failover** branch to display configured peers. When selecting a peer, the **Signature** will appear on the right.

- Enter **VLAN IP** addresses for each VLAN -- these are the IP addresses that you will assign later on the Equalizer running EQ/OS 10.

Note - The **Failover Parameters** section should display the **Failover IP** addresses you configured on the **VLAN Configuration** tab earlier.]

e. Click **Commit.**

f. Open the **Failover > Synchronization** tab and do the following:

- Check (turn on) the **dont transfer** check box.
- Uncheck (turn off) the **Use SSL only** check box.
- Click **Commit.**

g. Click **Help > About** and expand the system information box; the **failover mode** should now be **Primary** (you may need to manually refresh the browser to see the change of state immediately).

   If the failover mode remains in the **initializing** state, make sure that at least one server is active (that is, marked "up" in the GUI). Failover in EQ/OS 8.6 will not properly initialize if Equalizer cannot successfully probe at least one server on the network.

4. Configure the EQ/OS 10 system as the preferred backup:

   a. Configure all required VLANs, servers, and server pools as described in "Load Balancing & Networking" on page 87, "Configuring Server Connections" on page 767, and "Managing Server Pools" on page 476, using settings that are equivalent to the settings used on the EQ/OS 8.6 system.

Note - **Do not configure any clusters at this time**. Clusters wil be configured *after* configuring the system into failover, so that we don't have both systems trying to assign the same cluster IP addresses.

   Be sure to use the same VLAN IP addresses on the EQ/OS 10 system that you specified in Step 3d"Failover Between EQ/OS 8 and EQ/OS 10" on page 700on the EQ/OS 8.6 system.

   b. Assign a virtual IP address, the hb_interval and the heartbeat flags to the default subnet of each already configured VLAN, as in the examples below. The **virt_addr** values entered must match the **Failover IP** addresses and **Failover Netmasks** assigned on the EQ/OS 8.6 system in Step 2b. The heartbeat flag is used on the EQ/OS 10 subnet that will be used to communicate with the EQ/OS 8.6 system.

```
eqcli > vlan Mgmt subnet subnet name virt_addr 172.16.0.210 hb_interval 10
flags heartbeat
eqcli > vlan Server subnet subnet name virt_addr 192.168.0.210 hb_interval
10 flags heartbeat
```

Note - Both systems need to be heartbeating on the same subnet for failover to occur. Verify that the heartbeat flag is configured on the subnet that will be used to detect a failover in the EQ/OS 8.6 system.

   c. Enter the peer level **retry_interval** to the EQ/OS 10 local peer. Configure a value of at least 2 times the value of whatever the **hb_interval** value is from step 4b.

```
eqcli > peer peer name retry_interval 20
```

Note - Note that the **retry_interval** value of '20' is two times the **hb_interval** of 10 set in step 4b.

   d. Enter the following command to configure the peer object for the EQ/OS 8.6 system:

```
eqcli > peer eq_os8 signature os8-signature os8_intip os8_internal_addr
flags os8
```

You can copy the ***os8-signature*** from the EQ/OS 8.6 GUI by clicking the icon at the top of the left-frame tree to open up the **Failover > Required** tab. Copy the **Signature** value from the **This Equalizer** group at the top of the tab. If the EQ/OS 8.6 system is in dual network mode, replace ***os8_internal_ addr*** with the VLAN IP of the Equalizer running EQ/OS 8.6 on the non-Default VLAN. If is in single network mode, then specify the VLAN IP of the only defined VLAN.

e. Enable the **preferred_primary** and **failover** flags for the remote (EQ/OS 8.6) peer.

```
eqcli > peer eq_os8 flags preferred_primary,failover
```

f. Enable the **failover** flag and disable the **fo_config_xfer** flag for the local EQ/OS 10 peer.

```
eqcli > peer eq_os10 flags failover
eqcli > peer eq_os10 flags !fo_config_xfer
```

g. Display statistics for both peer objects using the **stats** command to confirm that failover has been negotiated between them, as in these examples:

Display the remote EQ/OS 8 peer statistics using this command:

```
eqcli > peer eq_os8 stats
```

```
    12200442: Peer eq_os8: Failover Group Status
    12200443: Member of Failover Group : Yes
    12200444: Preferred Primary : Yes
    12200445: Peer OS : EQ OS/8
    12200446: State : Probing
    12200447: Substate : Start
    12200448: Takeover : Primary Mode
    12200451: Last probe sent to this Peer : #2 at Fri Jan 7 22:03:40 2011
    12200452: Last probe received from this Peer: #2 at Fri Jan 7 22:03:41 2011
    12200453: Number of interfaces : 2
    12200456: Interface : Mgmt
    12200457: State : Probing
    12200458: Substate : Start
    12200459: Last probe sent on this if : #1 at Fri Jan 7 22:03:40 2011
    12200460: Last probe received in this if: #1 at Fri Jan 7 22:03:41 2011
    12200461: Number of strikes : 1
    12200456: Interface : Mgmt
    12200457: State : Probing
    12200458: Substate : Start
```

```
12200459: Last probe sent on this if : #1 at Fri Jan 7 22:03:40 2011
12200460: Last probe received in this if: #1 at Fri Jan 7 22:03:41 2011
12200461: Number of strikes : 1
```

Look carefully at the output for any errors. If you see any, or if the State is anything other than Probing:

- Check the VLAN configurations on both systems to ensure they are exactly the same, and correct if not. If this is the source of the issue, failover will begin to work as soon as the VLAN configurations match.
- Check the logs on both units for errors.

List the local EQ/OS 10 peer statistics using the command:

```
eqcli > peer eq_00:30:48:d3:ee:b6 stats

12200442: Peer eq_00:30:48:d3:ee:b6: Failover Group Status
12200443: Member of Failover Group : Yes
12200444: Preferred Primary : No
12200445: Peer OS : EQ OS/10
12200446: State : Probing
12200447: Substate : Start
12200448: Takeover : Backup Mode
12200449: Last Peer probed : eq_os8
12200450: Last Peer probed from : eq_os8
12200453: Number of interfaces : 2
12200456: Interface : Mgmt
12200457: State : Probing
12200458: Substate : Start
12200461: Number of strikes : 0
12200456: Interface : Mgmt
12200457: State : Probing
12200458: Substate : Start
12200461: Number of strikes : 0
```

h. Create all clusters using settings equivalent to those in use on the EQ/OS 8.6 system, using the commands described in the *EQ/OS 8.6 Installation and Administration Guide*.

i. Since the EQ/OS 10 Equalizer is in Backup Mode, it will not attempt to assume the cluster IP addresses until a failover occurs.

5. Set the **hb_interval** parameter (**Heartbeat Interval** in the GUI) on the EQ/OS 10 (local) peer to a value that is the Probe Interval parameter on the EQ/OS 8.6 system *times* the number of interfaces (VLANs) configured on the EQ/OS 8.6 system. For example if the EQ/OS 8.6 Probe Interval value is "5" and there are 2 VLANs configured, set the **hb_interval** parameter to "10" on the EQ OS 10 system.
6. Reboot the EQ/OS 8.6 system to make the EQ/OS 10 system the primary unit: open the **Equalizer > Maintenance** tab and click the **Reboot** button.
7. Once the EQ/OS 10 system detects that the EQ/OS 8.6 system is not responding to failover probes, the EQ/OS 10 peer configuration should indicate a change to **Primary Mode**. Use the command:

```
eqcli > peer eq_00:30:48:d3:ee:b6 stats

12200442: Peer eq_00:30:48:d3:ee:b6: Failover Group Status
12200443: Member of Failover Group : Yes
12200444: Preferred Primary : No
12200445: Peer OS : EQ OS/10
12200446: State : Probing
12200447: Substate : Start
12200448: Takeover : Primary Mode
12200449: Last Peer probed : eq_os8
12200450: Last Peer probed from : eq_os8
12200453: Number of interfaces : 2
12200456: Interface : Mgmt
12200457: State : Probing
12200458: Substate : Start
12200461: Number of strikes : 0
12200456: Interface : Mgmt
12200457: State : Probing
12200458: Substate : Start
12200461: Number of strikes : 0
```

The failover configuration is now complete, and the EQ/OS 10 system should have now assigned the cluster IP addresses to itself. The EQ/OS 8.6 GUI should display **initializing** for **failover mode** on the **Help > About** screen. Because of how failover in EQ/OS 8.6 works, the V8.6 unit will never transition to the **backup** failover state -- this is normal when an V8.6 Equalizer is configured in failover with a V10 Equalizer. It will remain in the **initializing** state until a failover occurs (while Equalizer is in **initializing** mode, it will not assume the cluster IP addresses).

The Equalizer running EQ/OS 10 will serve client requests and the Equalizer running EQ/OS 8.6 will remain in **initializing** mode. The two units will continuously exchange heartbeats with one another. Should the heartbeat exchange on any interface fail three times and on all other interfaces once, then the Equalizer running EQ/OS 8.6 will attempt to assume **primary** mode and assign the cluster IP addresses in the configuration to itself. Once the EQ/OS 10 system is available again, it will assume **backup** failover mode.

You can force the units to switch failover modes by rebooting the current **primary** Equalizer.

Note that the coyote icons at the top of the left frame of the EQ/OS 8.6 GUI will not change to indicate when the EQ/OS 10 peer is in primary mode -- that is, the EQ/OS 10 system will always have the sitting coyote icon next to it. On EQ/OS 8.6, always use the **Help > About** screen or the Equalizer log (**Equalizer > Status > Event Log**) to check failover status.

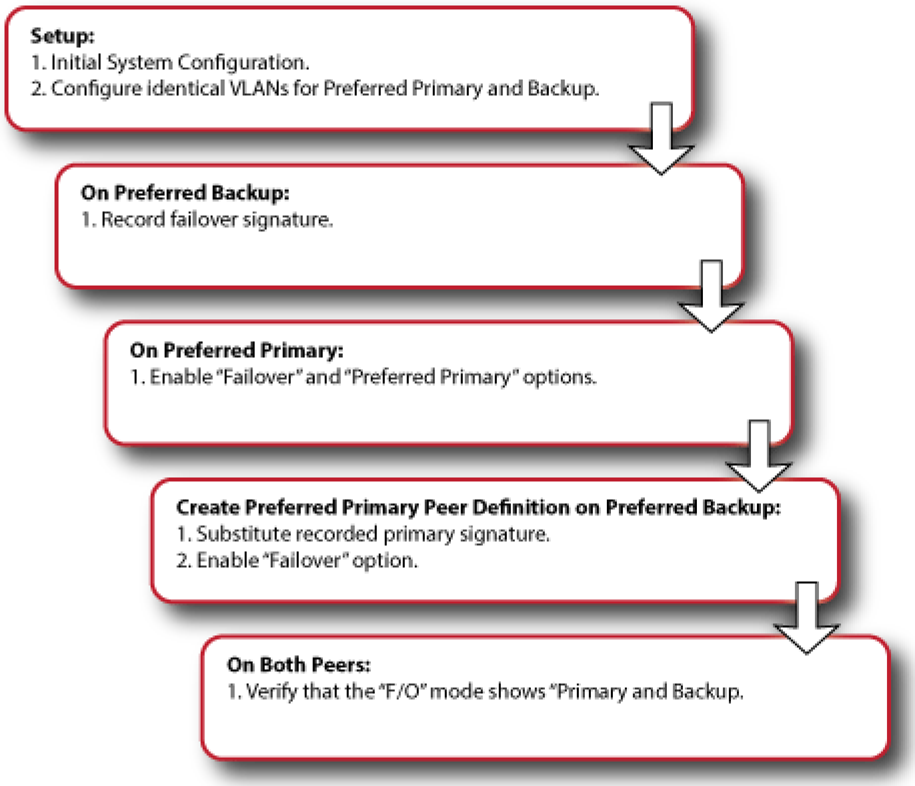
Configuring Active/Passive Failover Between Two Systems

When two Equalizers are configured into Active/Passive failover, they form a "failover pair". An Equalizer in a failover pair is called a "peer". At any given time, only one of the Equalizers in a failover pair is actually servicing requests sent to the cluster IP addresses defined in the configuration -- this unit is called the "active peer" or the "current primary" in the failover pair. The other Equalizer, called the “passive peer” or "current backup", does not process any client requests.

Both units continually send "heartbeat probes" or "failover probes" to one another. If the current primary does not respond to heartbeat probes, a failover occurs. In this scenario the current backup assumes the primary role by assigning the cluster IP addresses to its network interfaces and begins processing cluster traffic.

Configuration of failover peer definitions and options on VLAN subnets can be performed through the CLI or the GUI.

The figure below shows the suggested sequence of steps for enabling both Active/Passive Failover and the preliminary steps for configuring Active/Active and N+1 Failover.


Setup:
1. Initial System Configuration.
2. Configure identical VLANs for Preferred Primary and Backup.
On Preferred Backup:
1. Record failover signature.
On Preferred Primary:
1. Enable "Failover" and "Preferred Primary" options.
Create Preferred Primary Peer Definition on Preferred Backup:
1. Substitute recorded primary signature.
2. Enable "Failover" option.
On Both Peers:
1. Verify that the "F/O" mode shows "Primary and Backup.

Configuring VLAN (Subnet) Failover Settings (CLI)

Configure subnets for failover using the CLI as follows:

1. Configure VLANs and Subnets as described in "Configuring Subnets" on page 308. It is important that both the VLANs are identical in both the preferred primary and the backup.
2. Access the CLI as described in "Starting the CLI" on page 142.
3. Configure the failover service parameters for the preferred primary Equalizer:

Enter:

```
eqcli > vlan vlname vid vlan_id subnet sname flags flagname
```

Where ***vlname*** is the name of the VLAN,**vlan_id** is the VLAN ID number (A number between 1- 4094),***sname*** is the name of the subnet and ***flagname*** is the name of the flag.

The **Command** flag designates this subnet as the subnet over which the configuration file transfers (between preferred primary and preferred backup) can occur.

The **Heartbeat** flag allows the failover peers to probe one another over the subnet. At least one subnet must have a **Heartbeat** flag enabled.

4. Enter:

```
eqcli > vlan vlname subnet sname services service
```

Where ***vlname*** is the name of the VLAN, ***sname*** is the name of the subnet and ***service*** is the name of the service. These services enable the System Services for the Failover IP Address :

a. **fo_http** - when enabled the Equalizerwill listen for **http** connections on the Failover IP address on the subnet.

When configuring a Failover IP address on a subnet on Equalizer, verify that no other system on the network is using that Failover IP address, other than the Equalizers configured into failover.

If the IP address IS used by another system on the net, and a failover occurs, BOTH of the failover peers will transition to and remain in BACKUP mode.

1. **Remove the duplicate IP address from the offending system.**
2. **Reboot one of the peers.**

The peers should transition to their proper failover modes.

This applies to both Active/Passive and Active/Active failover.

a. **fo_https**- when enabled the Equalizer will listen for **https** connections on the Failover IP address on the subnet.

b. **fo_ssh** - when enabled **ssh** login will be permitted on the Failover IP address on the subnet.

c. **fo_snmp** - when enabled **snmp** will accept connections on the Failover IP address on the subnet.

d. **fo_envoy** - when enabled this will allow Envoy to monitor this subnet for failover

e. **fo_envoy_agent** - when enabled this will allow an Envoy agent to monitor this subnet for failover

5. Enter:

```
eqcli > vlan vlname subnet sname strike_count integer
```

Where ***vlname*** is the name of the VLAN, ***sname*** is the name of the subnet and ***integer*** is thee strike count number, which is the strike count threshold for a subnet. When the number of strikes detected on this subnet exceeds this value, the subnet has failed. A value of 0 indicates this subnet will never be considered "failed".

Note - If the **strike_count** reaches its specified maximum value on the subnet, or if the **strike_count** reaches 1 on all subnets in a multi-subnet network configuration, then a failover can occur. There is also a global**strike_count** (See "Failover Peer Probes and Timeouts" on page 695) in addition to the subnet **strike_count**. If the subnet **strike_count** is 0, then the global **strike_count** is used.

6. Enter:

```
eqcli > vlan vlname subnet sname hb_interval seconds
```

   Where ***vlname*** is the name of the VLAN, ***sname*** is the name of the subnet and ***seconds*** is the heartbeat interval or time in seconds (default: 2) between successful heartbeat checks of the peer.

7. Repeat the same procedure on the preferred backup.

Configuring VLAN (Subnet) Failover Settings (GUI)

Configure subnets for failover using the GUI as follows:

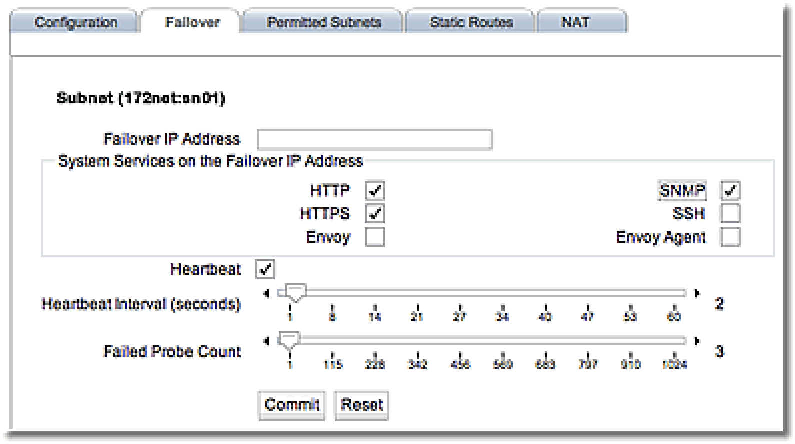
1. Configure both Equalizers running EQ/OS 10:

   a. Perform initial system configuration as outlined in "Load Balancing & Networking" on page 87.

   b. Create all required VLANs, clusters, servers, etc., required for your configuration.

   c. Ensure that the configuration is working properly. In particular, make sure that at least one server is active (that is, marked "up" in the GUI). Failover will not properly initialize if Equalizer cannot successfully probe at least one server.

2. Configure VLANs and subnets on both units; they must be exactly the same as noted above under"Failover Constraints" on page 690.
3. Designate a preferred primary and preferred backup using "Configuring Active/Passive Failover (CLI)" on page 714 beginning with step 3.
4. Open the Equalizer Graphical User Interface (or GUI) and click on the **System** configuration tab if it has not already been selected.Click on the arrows (▶) beside **Network** to continuously expand the branch until the subnet of the "Preferred Primary" is visible. Click on the subnet to display the subnet **Configuration** screen.
5. Click on the **Failover** tab to display the following. In the example shown below a subnet **sn01** for the VLAN **172net** is displayed.



6. Configure the failover parameters for the preferred primary Equalizer; in this case **sn01** on the VLAN **172net**. Use the check boxes and sliders as necessary. You will not be able to change the **Failover IP Address**. The **Failover IP Address** is used primarily as a server gateway and to provide an IP address for system services such as the GUI, SSH, etc.

When configuring a Failover IP address on a subnet on Equalizer, make absolutely sure that no other system on the network is using that Failover IP address other than the Equalizers configured into failover.

If the IP address IS used by another system on the net, and a failover occurs, BOTH of the failover peers will transition to and remain in BACKUP mode! The way to fix the problem is to:

1. Remove the duplicate IP address from the offending system.
2. Reboot one of the peers.

The peers should transition to their proper failover modes.

This applies to both Active/Passive and Active/Active failover.

7. Check the appropriate check boxes in the **Use Subnet IP Address** pane as follows:
   a. Checking the **Command Transfer** checkbox will designate this subnet as the subnet over which the configuration file transfers (between preferred primary and preferred backup) can occur.
   b. Checking the **Heartbeat** checkbox will allow the failover peers to probe one another over the subnet. At least one subnet must have a **Heartbeat** flag enabled.

Note - **Command Transfer** and **Heartbeat** use the subnet IP address, not the failover IP address.

8. Check the appropriate check boxes in the **Services on Failover IP Address** pane to select the allowable services that will be available:
   a. **HTTP** - when enabled the Equalizer will listen for **HTTP** connections on the Failover IP address on the subnet.
   b. **HTTPS**- when enabled the Equalizer will listen for **HTTPS** connections on the Failover IP address on the subnet.
   c. **SSH** - when enabled **SSH** login will be permitted on the Failover IP address on the subnet.
   d. **SNMP** - when enabled **SNMP** will accept connections on the Failover IP address on the subnet.
   e. **Envoy** -when enabled this will allow Envoy to monitor this subnet for failover
   f. **Envoy Agent** - when enabled this will allow an Envoy agent to monitor this subnet for failover

10. Adjust the **Heartbeat Interval** time in seconds (default: 2) between successful heartbeat checks of the peer.

11. Use the **Failed Probe Count** slider to adjust the number of failed peer probe attempts that must occur before marking a peer "down" (default: 3). If the **Failed Probe Count** reaches its specified maximum value on the subnet, or if the **Failed Probe Count** reaches 1 on all subnets in a multi-subnet network configuration, then a failover can occur. There is also a global **Failed Probe Count** (See "Failover Peer Probes and Timeouts" on page 695) in addition to the subnet **Failed Probe Count**. If the subnet **Failed Probe** count is 0, then the global **Failed Probe Count** is used.
12. Click on **Commit** when you have finished.
13. Perform Steps 1 through 11 above on the preferred backup.

Configuring Active/Passive Failover (CLI)

Perform Steps 1 and 2 on both Equalizers

1. Perform initial system configuration on both units as outlined in "Networking Technologies" on page 88.
2. Configure VLANs and subnets on both units; they must be exactly the same as noted in "Failover Constraints" on page 690 .

   Ensure that the network configuration is working properly. In particular, make sure that at least one gateway on a subnet that has the heartbeat flag enabled is responding to a ping command:

```
eqcli > ping gateway_IP_address
```

   If no gateways are responding, then configure a server with an IP address on a subnet with **heartbeat** enabled. Make sure it is responding to a **ping** command:

```
eqcli > server name proto {tcp|udp} ip IP_address port port_number
eqcli > ping server_IP_address
```

Perform Step 3 on the preferred backup Equalizer to obtain the peer signature:

3. Obtain the failover signature of the preferred backup Equalizer.
   a. Log in to the CLI on the Equalizer you will use as the preferred backup and enter:

```
eqcli show peer
------------------------------------
Configuration Sequence Number: 4593
------------------------------------
Peer Name               Type     Flags          F/O Mode        Error?
eq_001D7D78E13E (remote)OS/10    xfr            Standalone      No

Flags Key:
        F/O=> failover
        A/A=> active-active
        P/P=> preferred_primary
        xfr=> fo_config_xfer
        ssl => use_ssl
eqcli >
```

   b. Enter the **show peer** command again, however this time with the **Name** of the peer to display more details of the peer, including the signature.

```
eqcli > show peer name
```

Substitute the name of the peer displayed in the previous step for `Name`The information for that peer definition is displayed, as in this example:

```
eqcli > show peer   eq_001D7D78E13E
Peer Name              : eq_001D7D78E13E
Peer signature         :
1RBC78142F9ADE9E8F29FF5373AE7DA6EB994075A9BAAC1001B4
Peer sysid             : 00241DB2ABA0
Flags                  : failover, fo_config_xfer
[remainder of output omitted...]
```

c. Record the **`Peer signature`** displayed, or copy it using your terminal emulator’s supported editing commands. You’ll need it in the following steps.

Perform Steps 4 and 5 on the preferred primary Equalizer to add failover flags and to create a new peer definition for the backup.

You now need to configure the preferred primary Equalizer by adding failover flags and creating a peer on it for the backup that you created in steps 3 and 4. You will need the peer signature from the backup that you retained in step 4.

4. Log in to the Equalizer you will designate as the preferred primary and do the following:

   a. Display the peer name of the preferred primary by entering:

```
eqcli show peer
------------------------------------
Configuration Sequence Number: 4593
------------------------------------
Peer Name               Type     Flags          F/O Mode        Error?
eq_001D7D78E13E (local) OS/10    xfr            Standalone      No

Flags Key:
        F/O=> failover
        A/A=> active-active
        P/P=> preferred_primary
        xfr=> fo_config_xfer
        ssl => use_ssl
eqcli >
```

   b. Assign failover, **peferred_primary** flags to the preferred primary Equalizer by entering:

```
eqcli > peer name flags failover,preferred_primary,fo_config_xfer
```

   c. Verify that the flags are correct by entering the **show peer** command again to display the peer (preferred primary). The flags should display `F/O, P/P, xfr` beneath the `Flags` heading. The `fo_config_xfer` is used on the local peer and not on the remote peer. If it is enabled the primary peers on both systems will synchronized the configuration. When the flag is changed for the local peer, it should be reflected in the remote peer on the other system.

When the use_ssl flag is set, it causes messages from this Peer to a remote Peer to be transmitted using SSL. When not set, messages are transmitted in clear text.

The flag may be set differently for Peers in failover. For example, if set on Peer A, but not set on Peer B, heartbeats from Peer A to Peer B will be encrypted, however, heartbeats from Peer B to Peer A WILL NOT be. Also, a configuration synchronization request from Peer A to Peer B will be encrypted and so the response (Peer B's configuration) will also be encrypted. A configuration synchronization request, and the response, from Peer B to peer A will not be encrypted.

All transfers between a Peer that supports this flag and an older Peer that does not are done without using SSL, regardless of the setting of the flag on the Peer that supports it.

5. Now create a peer definition for the preferred backup on the primary Equalizer:

   a. Enter the following:

```
eqcli > peer name signature signature
```

   Substitute the signature of the preferred backup that you obtained in Step 3, above.

   b. Verify the new peer definition by entering:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: 4593
------------------------------------
Peer Name                  Type  Flags           F/O Mode        Error?
eq_00241DB2ABA0 (Local)   OS/10  F/O, P/P, xfr   Primary         No
eq_001D7D78E13E (Remote) OS/10  F/O,xfr         Backup          No

Flags Key:
        F/O => failover
        A/A => active-active
        P/P => preferred-primary
        xfr => fo_config_xfer
        ssl => use_ssl
```

   c. Now you will need the peer signature from the primary Equalizer. Enter the following:

```
eqcli > show peer name
```

   Where **name** is the name of the peer for the primary Equalizer. The following will be displayed.

```
eqcli > show peer   eq_001D7D78E13E
Peer Name              : eq_001D7D78E13E
Peer signature         :
1RBC14245CBCC7552362679F0E2AD4C0B2CF0C6E6B84AC1000D2
Peer sysid             : 001D7D78E13E
Flags                  : failover, fo_config_xfer
[remainder of output omitted...]
```

   Record the **Peer signature** displayed, or copy it using your terminal emulator’s supported editing commands. You’ll need it in the following steps.

Perform Step 6 on the preferred backup Equalizer to add failover flags and create a peer definition for the primary Equalizer.

6. Create a peer definition for the preferred primary, using the signature that you recorded in step 5:

   a. Display the peer name of the preferred backup by entering:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: 4593
------------------------------------
Peer Name               Type    Flags           F/O Mode        Message(s)
eq_00241DB2ABA(remote)  OS/10   xfr             Standalone      No

Flags Key:
        F/O => failover
        A/A => active-active
        P/P => preferred-primary
        xfr => fo_config_xfer
        ssl => use_ssl
```

   b. Add the failover flag to the backup by entering:

```
eqcli > peer name flags failover
```

   Where the peer **name** is the same one that appears beneath the **Peer Name** heading.

   c. Verify that the flag was assigned by entering:

```
eqcli > show peer
```

   d. Now create a peer definition for the preferred primary by entering the following:Create the peer definition for the preferred primary:

```
eqcli > peer name signature signature
```

   Substitute the signature of the preferred backup that you obtained in Step 5 for **signature**.

e. Verify the peer definitions by entering the following that should show the new peer definition:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: 4593
------------------------------------
Peer Name               Type    Flags           F/O Mode        Error
eq_00241DB2ABA0(Local)  OS/10   F/O, P/P, xfr   Primary         No
eq_001D7D78E13E(Remote)OS/10    F/O, xfr        Backup          No

Flags Key:
        F/O => failover
        A/A => active-active
        P/P => preferred-primary
        xfr => fo_config_xfer
        ssl => use_ssl
```

Note - Once the two peers are joined in a failover group (heartbeating and file sync are occurring), then they synchronize their remote peer definitions with the information obtained from the remote peer -The name and flags on the remote peer change. In addition, if you want to change the name of a peer, you MUST change the name of the local peer definition on that peer.

Perform Step 7 on both Equalizers.

7. Once both units start to communicate, displaying the peer definitions should indicate that the units have assumed the primary and backup failover roles.

   a. Confirm this on both units by entering:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: 4593
------------------------------------
Peer Name                       Type    Flags           F/O Mode        Error
eq_00241DB2ABA0 (Local) OS/10   F/O, P/P, xfr   Primary         No
eq_001D7D78E13E (Remote)        OS/10   F/O, xfr        Backup          No

Flags Key:
        F/O => failover
        A/A => active-active
        P/P => preferred-primary
        xfr => fo_config_xfer
        ssl => use_ssl
```

   Note that the **F/O Mode** column should appear as above when failover is working properly. The system on which you are logged in will always appear first in the list.

   b. Enter the following command for each peer listed:

```
eqcli > show peer name
```

   On each unit, the local peer definition (for the unit on which you are logged in) should appear like this example:

```
eqcli > show peer eq_00241DB2ABA0
Peer Name               : eq_00241DB2ABA0
Peer signature           : 1RBC78142F9ADE9E8F29FF5373AE7DA6EB994075A9BAAC1001B4
Peer sysid              : 00241DB2ABA0
Receive Timeout        : 2
Connect Timeout        : 1
Probe Interval         : 2
Retry Interval         : 5
Connectivity Retry Count: 8
Strike Count           :3
Flags                   : failover, preferred_primary, fo_config_xfer
OS/8 Internal IP        :
Number of Interfaces    : 2
  Member of Failover Group      : Yes
```

```
    Failover Enabled/Disabled     : Enabled
    Local/Remote Peer             : Local
    Preferred Primary             : Yes
    Peer OS                       : EQ/OS 10
    Peer State                    : Heartbeating:Start
    Failover State                : FOSM Complete:Idle
    Failover Mode         : Primary
    Last Peer heartbeated : eq_001D7D78E13E
    Last Peer heartbeated from    : eq_001D7D78E13E
    Interface                     : in1
        State                     : Heartbeating
         Substate         : Start
        Number of strikes : 0
        Subnet                    : Me2
           State                  : Heartbeating
           Substate               : Start
            Number of strikes     : 0
    Interface                     : in2
         State                    : Heartbeating
         Number of strikes        : 0
        Subnet                    : Me3
           State                  : Heartbeating
            Substate              : Start
            Number of strikes     : 0
```

The remote peer definition includes detailed information about the success or failure of the health check probes being sent by the local Equalizer (the unit on which you are logged in) to the remote Equalizer (the other peer).

Look carefully at the output for any errors. If you see any, or if the **`State`** is anything other than Probing on any interface (subnet) on which **`heartbeat`** is enabled:

- Check the VLAN configurations on both systems to ensure they are exactly the same, and correct if not. If this is the source of the issue, failover will begin to work as soon as the VLAN configurations match.
- Check the logs on both units for errors.

The remote peer display should appear like this example:

```
eqcli > show peer eq_001D7D78E13E
Peer Name                : eq_001D7D78E13E
Peer signature           : 1RBC14245CBCC7552362679F0E2AD4C0B2CF0C6E6B84AC1000D2
Peer sysid               : 001D7D78E13E
Receive Timeout         : 2
Connect Timeout         : 1
Probe Interval          : 2
Retry Interval          : 5
Connectivity Retry Count: 8
```

```
Strike Count           :3
Flags                   : failover, fo_config_xfer
OS/8 Internal IP        :
Number of Interfaces    : 2
  Member of Failover Group      : Yes
  Failover Enabled/Disabled     : Enabled
  Local/Remote Peer             : Remote
  Preferred Primary             : No
  Peer OS                       : EQ/OS 10
  Peer State                    : Heartbeating:Start
  Failover State                : FOSM Complete:Idle
  Failover Mode         : Backup
  Last heartbeat sent           : #322 at Wed Mar 14 12:07:10 2012
  Last heartbeat received       : #194 at Wed Mar 14 12:07:10 2012
  Interface                     : in1a
     State                      : Heartbeating
     Substate                   : Start
     Last heartbeat sent        : #161 at Wed Mar 14 12:07:10 2012
      Last heartbeat received   : #97 at Wed Mar 14 12:07:10 2012
      Number of strikes : 0
     Subnet                     : Me1
         State                  : Heartbeating
         Substate               : Start
        Last heartbeat sent     : #161 at Wed Mar 14 12:07:10 2012
         Last heartbeat received : #97 at Wed Mar 14 12:07:10 2012
         Number of strikes      : 0
  Interface                     : in2
      State                     : Heartbeating
      Substate                  : Start
     Last heartbeat sent        : #161 at Wed Mar 14 12:07:10 2012
      Last heartbeat received   : #97 at Wed Mar 14 12:07:10 2012
     Number of strikes          : 0
      Subnet                    : sn1
        State                   : Heartbeating
         Substate               : Start
        Last heartbeat sent     : #161 at Wed Mar 14 12:07:10 2012
         Last heartbeat received : #97 at Wed Mar 14 12:07:10 2012
         Number of strikes      : 0
```

The above display includes detailed information about the success or failure of the health check probes being sent by the remote Equalizer (the other peer) to the local Equalizer (the unit on which you are logged in).

Refer to "Peer Interface Subnet States and Substates" on page 697 for descriptions of the Peer states and substate conditions.

Displaying the Failover Summary.

You can display the failover summary by entering `eqcli > show failover`. The following is an example of the failover summary:

```
eqcli > show failover
Local Peer Failover Information
Command subnet:         Vlan v2, Subnet sn172
Failover:               Enabled
Mode:                   Active/Passive
Preferred Primary:      Yes
Config Sync:            Enabled
eqcli >
```

Using the "Rebalance" Command

If the preferred primary peer is in "Backup" mode, and you enter `eqcli > rebalance`, the preferred primary peer will take over as the Primary peer.

Configuring Active/Passive Failover (GUI)

Perform Steps 1 and 2 on both Equalizers.

1. Perform initial system configuration on both units as outlined in Networking Technologies.
2. Configure VLANs and subnets on both units; they must be exactly the same as noted in Failover Constraints.

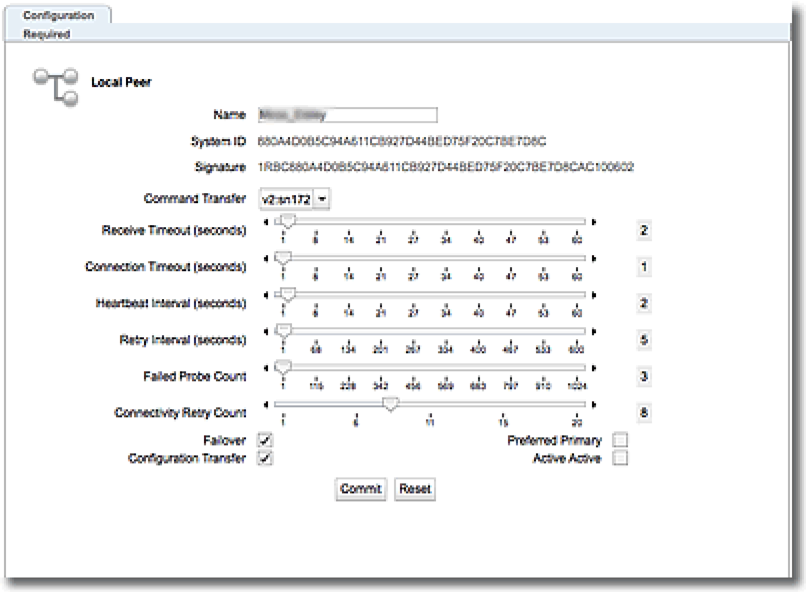
Perform Step 3 on the preferred backup Equalizer to obtain the peer signature.

3. Log in to the GUI for the backup Equalizer using the procedures described in "Logging In" on page 238.

    a. Click on the **System** configuration tab on the left navigational pane if it is not already selected.

    b. Click on the arrow (▶) beside failover to expand the branch.

    c. Click on a peer to be used as the preferred backup. The following will be displayed:



Configuration Required

Local Peer

Name [redacted]

System ID 880A4D0B5C94A611CB927D44BED75F20C7BE7D8C

Signature 1RBC880A4D0B5C94A611CB927D44BED75F20C7BE7D8CAC100602

Command Transfer v2:sn172

Receive Timeout (seconds) 2

Connection Timeout (seconds) 1

Heartbeat Interval (seconds) 2

Retry Interval (seconds) 5

Failed Probe Count 3

Connectivity Retry Count 8

Failover ✓ Preferred Primary ☐

Configuration Transfer ✓ Active Active ☐

    d. Check the **Failover** flag.

    e. Highlight and copy the failover **Signature**of the preferred backup Equalizer. Copy the signature to an electronic clipboard, notepad or whatever means available to save it.

Perform Steps 4 and 5 on the preferred primary Equalizer to add failover flags and to create a new peer definition for the backup.

You now need to configure the preferred primary Equalizer by adding failover flags and creating a peer on it for the backup that you configured in steps 3 and 4. You will need the peer signature from the backup that retained in step 4.

4. Log in to the GUI for the preferred primary Equalizer using the procedures described in "Logging In" on page 238.

   a. Configure the preferred primary peer and check the **Failover**and **preferred_ primary**flags to the preferred primary Equalizer as shown below.

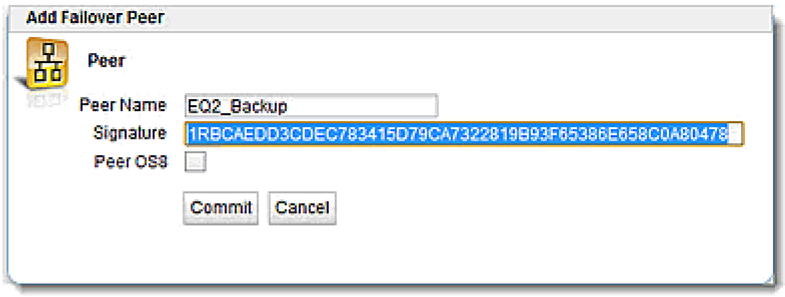
   b. Highlight and copy the failover **Signature**of the preferred primary Equalizer. Copy the signature to an electronic clipboard, notepad or whatever means available to save it.

   c. Configure the timeout and interval sliders using the descriptions provided in "Failover Peer Probes and Timeouts" on page 695.

   d. Click on **Commit** to save the flag assignments.

5. Create a peer definition for the backup peer- on the preferred primary Equalizer.

   a. Right click on **Failover** on the left navigational pane and select **Add failover peer.** The failover peer entry form as shown below:



Add Failover Peer

Peer

Peer Name: EQ2_Backup

Signature: 1RBCAEDD3CDEC783415D79CA7322819B93F65386E658C0A80478

Peer OS8: ☐

   b. Enter a **Peer Name** for the backup peer and type or paste in the signature of the preferred primary that you saved from step 3. Click on **Commit** to save the peer.

   c. You now need to assign a failover flag to the backup peer. Click on the backup peer (**EQ2_Backup** in the example) to display the backup peer **Configuration > Required** screen.

   d. Enable the **Failover** flag and click on **Commit**. Both peers should appear on the left navigational pane on the **Peers** branch.

Perform Step 6 on the preferred backup Equalizer to add failover flags and create a peer definition for the primary Equalizer.

6. Log back in to the GUI for the backup Equalizer using the procedures described in "Logging In" on page 238. You will need to create a peer for the preferred primary, using the signature that you recorded from step 4b.

   a. Right click on **Failover** in the left navigational pane and select **Add failover peer** to display the peer entry form .

   b. Type or paste the peer Signature of the preferred primary Equalizer and click on **Commit**

c. Configure the timeout and interval sliders using the descriptions provided in "Failover Peer Probes and Timeouts" on page 695.

c. Enable the Failover flag and click on **Commit**.

Note - Once the two peers are joined in a failover group (heartbeating and file sync are occurring), then they synchronize their remote peer definitions with the information obtained from the remote peer -The name and flags on the remote peer change. Once the two peers are joined in a failover group (heartbeating and file sync are occurring), then they synchronize their remote peer definitions with the information obtained from the remote peer -The name and flags on the remote peer change. In addition, if you want to change the name of a peer, you MUST change the name of the local peer definition on that peer.

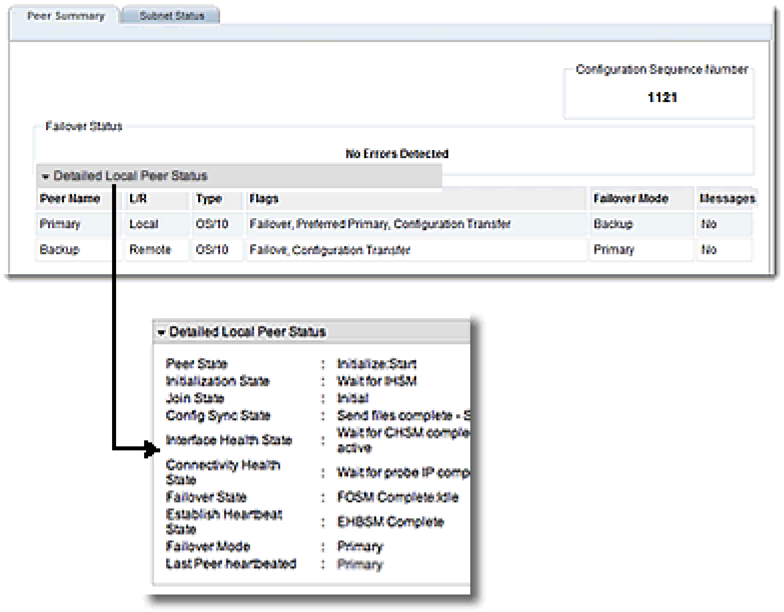
Perform Steps 7 on both Equalizers.

You have now configured failover peers in both the preferred primary and backup Equalizers. To verify that you have correctly configured failover do the following.

7. Access the GUI for the preferred primary or backup Equalizer.

   a. Click on the **System** configuration tab if it is not already selected.

b. Click on **Peers**on the left navigational pane to display the Peers summary screen as shown below. Note that since the first screen shows the preferred primary Equalizer**Peer Summary**as it is categorized as Local and if a failover state exists it will become the backup. Click on **Detailed Local Peer Status** for drop down box with detailed peer information.



Configuration Sequence Number
1121

Failover Status
No Errors Detected

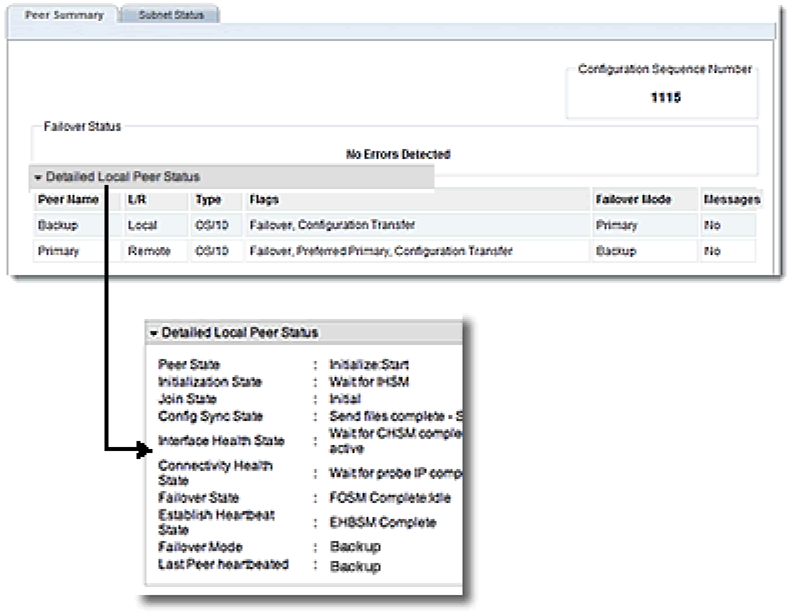
▾ Detailed Local Peer Status

Peer Name	L/R	Type	Flags	Failover Mode	Messages
Primary	Local	OS/10	Failover, Preferred Primary, Configuration Transfer	Backup	No
Backup	Remote	OS/10	Failove, Configuration Transfer	Primary	No

▾ Detailed Local Peer Status

Peer State	:	Initialize:Start
Initialization State	:	Wait for IHSM
Join State	:	Initial
Config Sync State	:	Send files complete - S
Interface Health State	:	Wait for CHSM comple active
Connectivity Health State	:	Wait for probe IP comp
Failover State	:	FOSM Complete:Idle
Establish Heartbeat State	:	EHBSM Complete
Failover Mode	:	Primary
Last Peer heartbeated	:	Primary

The following shows the preferred backup **Equalizer Peer Summary** and shows the reversed condition in a failover state. Click on **Detailed Local Peer Status** for drop down box with detailed peer information.



Configuration Sequence Number
1115

Failover Status
No Errors Detected

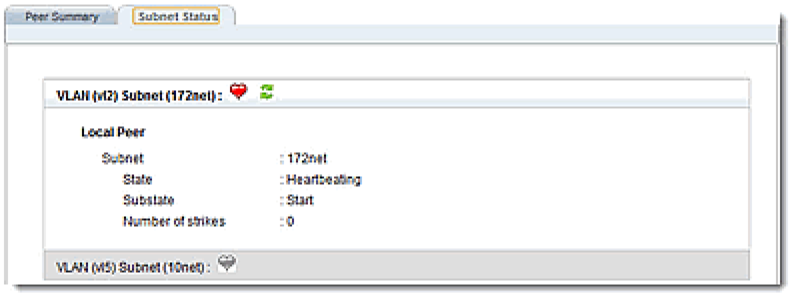
▾ Detailed Local Peer Status

Peer Name	L/R	Type	Flags	Failover Mode	Messages
Backup	Local	OS/10	Failover, Configuration Transfer	Primary	No
Primary	Remote	OS/10	Failover, Preferred Primary, Configuration Transfer	Backup	No

▾ Detailed Local Peer Status

Peer State	:	Initialize:Start
Initialization State	:	Wait for IHSM
Join State	:	Initial
Config Sync State	:	Send files complete - S
Interface Health State	:	Wait for CHSM comple active
Connectivity Health State	:	Wait for probe IP comp
Failover State	:	FOSM Complete:Idle
Establish Heartbeat State	:	EHBSM Complete
Failover Mode	:	Backup
Last Peer heartbeated	:	Backup

c. You can view the subnet stats of each by selecting the Subnet Status tab for each showing a **Heartbeating**condition. The first is the preferred primary load balancer and the second is the backup.



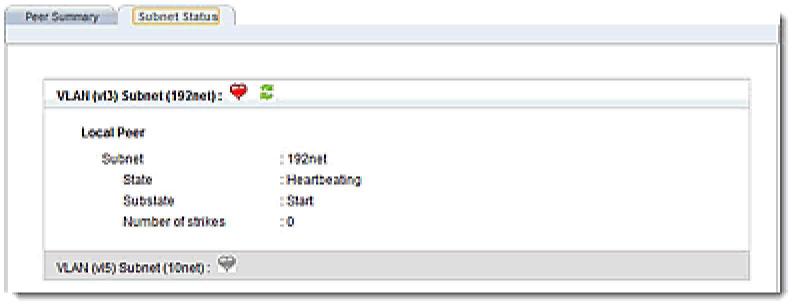
Peer Summary | Subnet Status

VLAN (vl2) Subnet (172net) :

Local Peer

Subnet	: 172net
State	: Heartbeating
Substate	: Start
Number of strikes	: 0

VLAN (vl5) Subnet (10net) :



Peer Summary | Subnet Status

VLAN (vl3) Subnet (192net) :

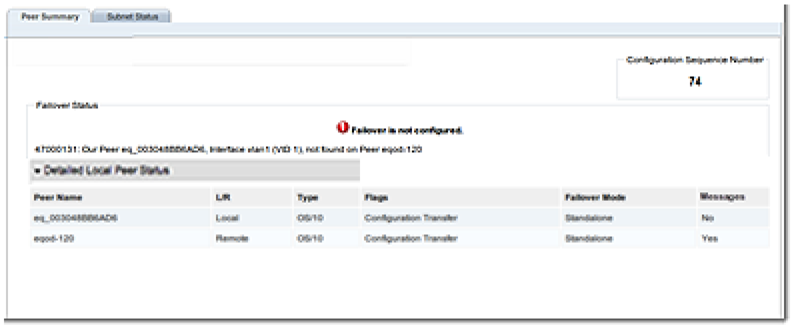
Local Peer

Subnet	: 192net
State	: Heartbeating
Substate	: Start
Number of strikes	: 0

VLAN (vl5) Subnet (10net) :

Peer Summary Display Showing Errors

If failover were NOT configured correctly or a problem existed with one of the peers, you would see a display similar to the following example. Note that a failure icon (❗) appears on the left navigational pane beside the peer with an error as well as on the right indicating that **Failover is not configured.** :



Peer Summary | Subnet Status

Configuration Sequence Number
74

Failover Status

Failover is not configured.

47000131: Our Peer eq_003048B86AD6, Interface vlan1 (VID 1), not found on Peer eqod-120

▾ Detailed Local Peer Status

Peer Name	L/R	Type	Flags	Failover Mode	Messages
eq_003048B86AD6	Local	OS/10	Configuration Transfer	Standalone	No
eqod-120	Remote	OS/10	Configuration Transfer	Standalone	Yes

Refer to Peer, Interface, Subnet States and Substates for descriptions of the Peer states and substate conditions.

Configuring Active/Active Failover

Active/Active (A/A) failover allows clusters to be active on both Peers that are configured into failover. For the same failure situations that cause a Peer to take over all the cluster and floating IP addresses in an Active/Passive failover configuration, Active/Active failover operates the same way - that is that the healthy Peer will take over all of the cluster and failover IPs.

An Active/Active failover configuration consists of two peers. Equalizer’s clusters are instantiated on both peers and organized into “Failover Groups”. If the one peer’s connectivity for the failover group’s resources is judged to be “healthier” than the peer on which the group is running, then the group “fails over” to the other peer.

It should be noted that if and when the "sick" Peer is healed, there is no automatic migration of the clusters back to it. You can, however, invoke a "rebalance" command to make this happen.

Failover Groups

Active/Active failover introduces the concept of "Failover Groups". A Failover Group consists of all the smallest set of resources that may be moved between Peers and can consist of one or more clusters, servers, and failover IPs.

Failover Groups are dynamically determined by the configuration and cannot be specified by the user.

In the simplest case, there is a maximum of 1 Failover group per subnet. However, based on the cluster/match rule/server pool/server configuration, a Failover Group may contain more than one subnet. Basically, the algorithm is that:

- a cluster subnet and, for *non-Spoof* match rules, all associated server subnets must all be in the same Failover Group. This means that all clusters and failover IPs on any of these subnets are in the same Failover Group.
- if *spoof* is set for all the cluster match rules, then the server subnets are not factored into the Failover group.

This means that as clusters and servers are added or deleted, the Failover group configuration may change.

Configuring Active/Active Failover (CLI)

1. Configure failover using the current procedures described in "Configuring Active/Passive Failover (CLI)" on page 714.
2. Activate the Active/Active failover mode by setting the active-active flag on the local Peers. This flag must be set on both Peers for A/A to be enabled. If the flag is set on only one, or no Peers, failover operates as in the current Active/Passive mode.

You will need to access both Equalizers so it may be easier to open two TTY sessions. Each one should access the local peers. Enter the following for each local peer:

```
eqcli > peer [name] flags active-active
```

Once you have added active-active flags to each local peer if the Equalizers are heartbeating you should see the A/A flags should be displayed when you enter show peer for each Equalizer as shown below. One Equalizer should be displayed as “Backup” while the other as “Primary”.

- If all Failover groups are instantiated on a Peer, the F/O column will display Primary.
- If none are instantiated, the F/O column will display Backup.
- If some Failover groups are instantiated on one Peer and some on the other, the F/O column will display Mixed.

Primary Equalizer

```
eqcli > show peer

Peer Name          Type   Flags             F/O Mode      Message(s)

Primary (Local)    OS/10  F/O,A/A,P/P,xfr   Backup        No
Backup (Remote)    OS/10  F/O,A/A,xfr       Primary       No

Flags Key:
        F/O => failover
        A/A => active-active
        P/P => preferred-primary
        xfr => fo_config_xfer
        ssl => use_ssl

eqcli >
```

Backup Equalizer

```
eqcli > show peer

Peer Name          Type  Flags          F/O Mode        Message(s)

Backup (Local)     OS/10 F/O,A/A,xfr    Primary         No
Primary (Remote)   OS/10 F/O,A/A,P/P,xfr Backup         No

Flags Key:
        F/O => failover
        A/A => active-active
        P/P => preferred-primary
        xfr => fo_config_xfer
        ssl => use_ssl

eqcli >
```

3. As indicted previously Failover Groups are dynamically determined by the configuration and cannot be specified by the user. As clusters and servers are added or deleted, the Failover group configuration may change.
4. Set the **preferred_peer** flag on a cluster. The purpose of the **preferred_peer** parameter is to indicate the failover peer on which the cluster is "desired" to run, and it is the peer on which the cluster will be run if the user runs the rebalance command. This parameter is set on the Peer that your want the cluster to be associated with, in the non-failover case. If this parameter is not set, the cluster defaults to the Peer that has been set as the preferred primary.

When the use_ssl flag is set, it causes messages from this Peer to a remote Peer to be transmitted using SSL. When not set, messages are transmitted in clear text.

The flag may be set differently for Peers in failover. For example, if set on Peer A, but not set on Peer B, heartbeats from Peer A to Peer B will be encrypted, however, heartbeats from Peer B to Peer A WILL NOT be. Also, a configuration synchronization request from Peer A to Peer B will be encrypted and so the response (Peer B's configuration) will also be encrypted. A configuration synchronization request, and the response, from Peer B to peer A will not be encrypted.

All transfers between a Peer that supports this flag and an older Peer that does not are done without using SSL, rregardless of the setting of the flag on the Peer that supports it.

Note - The **preferred_peer** flag for all clusters that would be configured into the same Failover Group must be the same.

```
eqcli > cluster [name] flags preferred_peer
```

Show the cluster parameters by entering :

```
eqcli > show cluster cl-tcp

L4 Cluster Name  : cl-tcp
Protocol         : tcp
IP Address       : 172.16.0.131
Port             : 80
Port Range       : 0
Preferred Peer   : Primary
VID              : 1
Server Pool      : testserverpool
Sticky Timeout   : 5
Sticky Netmask   : 0
Idle Timeout     : 5
Stale Timeout    : 5
L4 Flags         :
eqcli >
```

5. Display the dynamically created failover group by entering **`show fogrp`**. This command lists the Failover Groups that have been created, based on the configuration. For example:

```
eqcli > show fogrp

F/O Group Name          F/O Group ID    F/O Mode        Primary Peer
Unassigned              0               Not Used
fo_group1               1               Primary         Primary
fo_group2               2               Backup
```

The `F/O Group Name` “Unassigned” is used:

- When active-active is NOT enabled on the local peer, all clusters are in the "Unassigned" F/O Group.
- If the system cannot determine a failover group in which to place a cluster, possibly because it was assigned a VLAN IP that does not reside on a defined subnet, it goes into Unassigned.

Display the elements of the failover group by entering show **fogrp <*name*>** - where **<*name*>** is one of the names in the list. For example:

```
eqcli > show fogrp fo_group2

F/O Group fo_group1:
   ID                    = 1
   Preferred Peer        = Primary
   Primary Peer          = Backup
   F/O Mode              = Backup
Subnet Members (num = 1):
   God:Me
Cluster Members (num = 2):
   cl01
   cl02
No Server Members
```

Note - At least 2 subnets must be configured, along with their failover IP addresses.

6. When the "sick" Peer has been "healed", the clusters do not automatically migrate back. For this to occur you will need to invoke the rebalance command to make this happen. This command is at the global context level and causes the clusters to be "moved back" to their preferred peer. Enter: `eqcli >` **`rebalance`**

"Rebalance" Command Notes

1. When a cluster is added:
   a. If a preferred peer has not been set for the cluster,the system checks whether the cluster should belong to an existing F/O Group--based on its IP address. If it does, its preferred peer is set as the preferred peer of that F/O Group-- otherwise, it is set as the preferred primary.
   b. If the preferred peer has been set on the cluster, the system checks to see whether the cluster can be added to a F/O Group with the same preferred peer. If not, the request is rejected and an error message will be generated.
   c. A newly added cluster is not rebalanced until the rebalance command is used.
2. If a cluster's preferred peer has been changed, the system checks whether the cluster can be added to a F/O Group with the same preferred peer. If not, the request is rejected and an error message will be generated.
3. If the active-active flag is set on all peers, and the rebalance command is executed, the rebalancing operation will function correctly.
4. If multiple peers think they own a F/O Group, the peer set as the preferred peer will be the preferred primary.

Testing Active/Active Failover

1. First verify that Active/Passive failover works. Refer to "Enabling Active/Passive Failover Between Two Systems" on page 707.
2. With Active/Active failover enabled:
   a. Configure more than 1 cluster, but all in the same Failover Group and verify that failover works the same as with an Active/Passive setup. Perform this with preferred_peer not set on the clusters, and then set and verify that the clusters are instantiated on the appropriate Peer and that they "failed over" as expected.
   b. Configure one or more clusters on each subnet, setting a different preferred_ peer for each Failover Group. Verify that the clusters are instantiated on the appropriate Peer and that they "fail over" as expected.
   c. Using multiple failover groups, configure servers for the clusters, using both spoof and non-Spoof.For non-spoof, configure servers on a different subnet than the cluster. For spoof, configure servers on a different subnet than the cluster.
   d. Using multiple Failover Groups, change the configuration such that 2 Failover groups will be merged and verify that all work as expected. For example, suppose there are 2 F/O Groups:
      - F/O Group 1 -has subnet 172.16.0/24 with cluster cl01 (172.16.0.211), server sv01 (172.16.0.181) and floating IP 172.16.0.219.
      - F/O Group 2 - has subnet 192.168.0/24 with cluster cl02 (192.168.0.211), server sv02 (192.168.0.181) and floating IP 192.168.0.219.
      - If the clusters are using spoof and sv03 (192.168.0.182) is added to the server pool for cluster cl01, this will cause F/O Groups 1 and 2 to be merged into a single F/ O Group that includes cl01, cl02, sv01, sv02, sv03, and subnets 172.16.0/24 and 192.168.0/24.
   e. Then, change the configuration such that the single F/O group is split into 2 F/O Groups.(e.g., by deleting sv03 from cl01, above.)

Verify that:

- All clusters can pass traffic.
- Failover occurs as expected if one Peer fails.
- The F/O Groups are rebalanced appropriately when the rebalance command is executed.

Configuring N+1 Failover

N+1 Failover is a feature where the failover configuration consists of multiple active peers ("N") plus 1 passive peer. In this type of failover configuration, the Equalizer clusters are instantiated on all "N" peers and organized into failover groups. If the passive, or backup peer's connectivity for a failover group's resources is judged to be "healthier" that the peer on which the group is running, then the group fails over to the passive peer, which becomes the Primary peer.

N+1 failover provides the ability to configure up to 4 Peers in a failover configuration. All peers will be "heartbeating" with each other and all synchronizing configuration. If a Peer "fails", for each failover group (F/O) group affected by the failure, one and only one of the other Peers will take over the F/O Group based on:

1. Which Peer in the F/O group has the best connectivity to servers, routers, etc. In case of a tie amongst one or more Peers, the Peer with the greatest System ID hex value or " sysid" will take over the F/O group. For example, if 2 peers have the same level of connectivity with servers, routers, etc, and one "sysid" is "003048BC2C8A" and the other is "003048D52AA2". The second "sysid" has a higher hex value and will take over the F/O group.

Note - All flavors of Microsoft© Windows include a hex calculator. A free, downloadable Hex Calculator widget for Mac OX is also available.

2. The user can subsequently adjust the load across the Peers by managing the preferred Peer for each F/O Group and executing the rebalance function.

Note - Currently, if a failover event occurs, all F/O Groups are moved when a failover event occurs, even if it only affects a subset of the F/O Groups. Failover occurs on a F/O Group basis. For example, if an interface goes down, only the affected F/O Group(s) will be moved.

Network Design for N+1 Failover

The design of the host network is critical to a successful failover configuration.

The essential concept of active-active failover is that resources that are required for a cluster to serve client requests are organized into "failover groups". For any cluster, the required resources include:

- the cluster object and all objects to which it points including server pools, server instances, servers, responders, certificates, etc.
- the subnet on which the cluster IP address resides
- the subnet (or subnets) on which all server IP addresses in the server pool reside

If you instead locate a cluster IP address on one subnet and the servers in the cluster's server pool all reside on another subnet, then both those subnets would be considered part of the cluster's failover group.

So, in order to allow each cluster to fail over separately to another Equalizer, the cluster IP address and all server IP addresses need to be located on Equalizer subnets that are distinct from the subnets on which other cluster and server IP addresses reside.

Once you configure cluster and server IPs and enable active-active failover, the clusters, servers, subnets, etc., are organized into "failover groups" that can be passed between all the peers at network connectivity issues occur.

How a Peer is Chosen for Failover in N+1 Configuration

A failover occurs when Equalizer detects that there is an issue with one of the subnets on which a cluster's IP address or one of its server IP addresses resides. This typically means that Equalizer has lost connectivity on a subnet, and can happen for any number of reasons; for example, the failure of a downstream hub, router, or other networking device.

A failover event can be simulated by either removing a cable from Equalizer's front panel or rebooting a peer.

In our example configuration, each VLAN subnet is connected to Equalizer through a separate port. When you remove a cable from Equalizer, it recognizes that it has lost connectivity on that subnet and attempts to fail over all the resources on that subnet (the "failover group") to another peer.

When Equalizer detects a network connectivity failure, it does the following:

1. It determines which failover groups are affected by the failure.
2. It examines the heartbeat information it has received from the other peers in the failover set, and determines which other peers can provide connectivity on the subnets that have failed.
3. If there is only one peer that can provide the required connectivity, the failover group is moved to that peer.
4. If there is more than one peer that can provide the required connectivity, Equalizer checks the 'preferred peer' setting on the cluster (or clusters) in the failover group (or groups), and if the preferred peer can provide connectivity, the failover groups are moved to that peer.

5. If the preferred peer is not one of the systems that can provide connectivity, or if a cluster has no preferred peer set, then Equalizer checks to see if the peer that has the ‘preferred primary’ flag set can provide the required connectivity. If it can, the failover groups are moved to that peer.
6. If the preferred primary is not one of the systems that can provide connectivity, the Equalizer checks the System ID of all the peers that can provide the required connectivity, and moves the failover groups to that peer.
7. If all the above fails to select a peer to which the failover groups can be moved, they remain instantiated on the current peer.

Note - In Step 6, above, the Equalizer System ID number is used to break "ties" if checking the preferred peer and preferred primary settings fail to identify a peer to which we can fail over, and there is more than one peer available that can provide the required connectivity. This is why the system with the highest System ID is used as the "+1" backup unit in all the sample configurations, so that we are guaranteed to move a failover group over to the dedicated backup unit when there is no preferred peer or preferred primary available that provides the connectivity required by the failover group.

Equalizer’s System ID is displayed in the CLI using the global context version command:

```
eqcli > version

Firmware Version                : 10.3.2d
Firmware O/S Tag                : AA

System Type                     : ADC
System Revision                 : 2
System Serial Number            : A08CA-16001
System ID                       : 0012345ABCDE

Features                        : Hardware SSL Acceleration
Switch Type                     : CP302
Switch Count                    : 1
Current Firmware ID             : 23
Latest Firmware ID              : 23

Support Information:
Last refresh date               :
Hardware Support End            :
Hardware Support Level          :
Firmware Support End            :
Firmware Support Level          :
Enhanced Support End            :
Enhanced Support Level          :
Email                           :

eqcli >
```

The System ID is a 12-digit hexadecimal number. You can use the hexadecimal setting on a calculator to determine which of your systems has the highest System ID.

Monitoring N+1 Failover

There are several CLI commands you can use to monitor failover status:

Displaying Failover Group Status

Failover groups are configured by Equalizer automatically according to your network topology and the subnets on which cluster and server IP addresses reside. You can modify the failover group configuration only by modifying your cluster IP addresses, server IP addresses, and subnet configuration.

To display the current list of failover groups, use the **`show fogrp`** global context command:

```
eqcli > show fogrp

F/O Group Name         F/O Group ID    F/O Mode        Primary Peer

Unassigned             0               Not Used
fo_group1              1               Primary         Eq-A
fo_group2              2               Backup          Eq-B
```

The four columns contain the following details information:

`F/O Group Name`	These are determined by Equalizer, according to cluster IP addresses, server IP addresses, and the network configuration. 'Unassigned' is the failover group used when active-active failover is not yet enabled. Failover groups are not used in active-passive failover configurations.
`F/O Group ID`	An identifying number for the failover group. This is set by Equalizer and not direcetly modifiable.
`F/O Mode`	Indicates whether the system on which you executed the command is the current Primary for this failover group, or whether the system is a Backup for this failover group. If a system is the Primary for this group, it instantiates all the cluster IP addresses and failover IP addresses necessary for the failover group.
`Primary Peer`	Shows the peer name of the Equalizer that is currently Primary for this failover group.

Detailed failover group status can be obtained by supplying a group name to the **show fogrp** command:

```
eqcli > show fogrp fo_group1

F/O Group fo_group1:
   ID                   = 1
   Perferred Peer       = Eq-A
   Primary Peer         = Eq-A
   F/O Mode             = Primary
   Subnet Members (num = 1):
        V12:172net
   Cluster members (num = 1):
        clA
   Server members (num = 1):
        sv2
```

In addition to the ID, peer, and mode information (see the previous table), this command displays exactly which subnets, clusters, and servers belong to this failover group. These are the objects that will become active on another peer when this failover group is moved as a result of a failover event.

This form of the command allows you to determine exactly how your clusters, servers, and subnets are organized by Equalizer into failover groups, and also which clusters are instantiated on which Equalizers at any given time.

Displaying Peer Status

The show peer command displays a summary of all the currently defined peers:

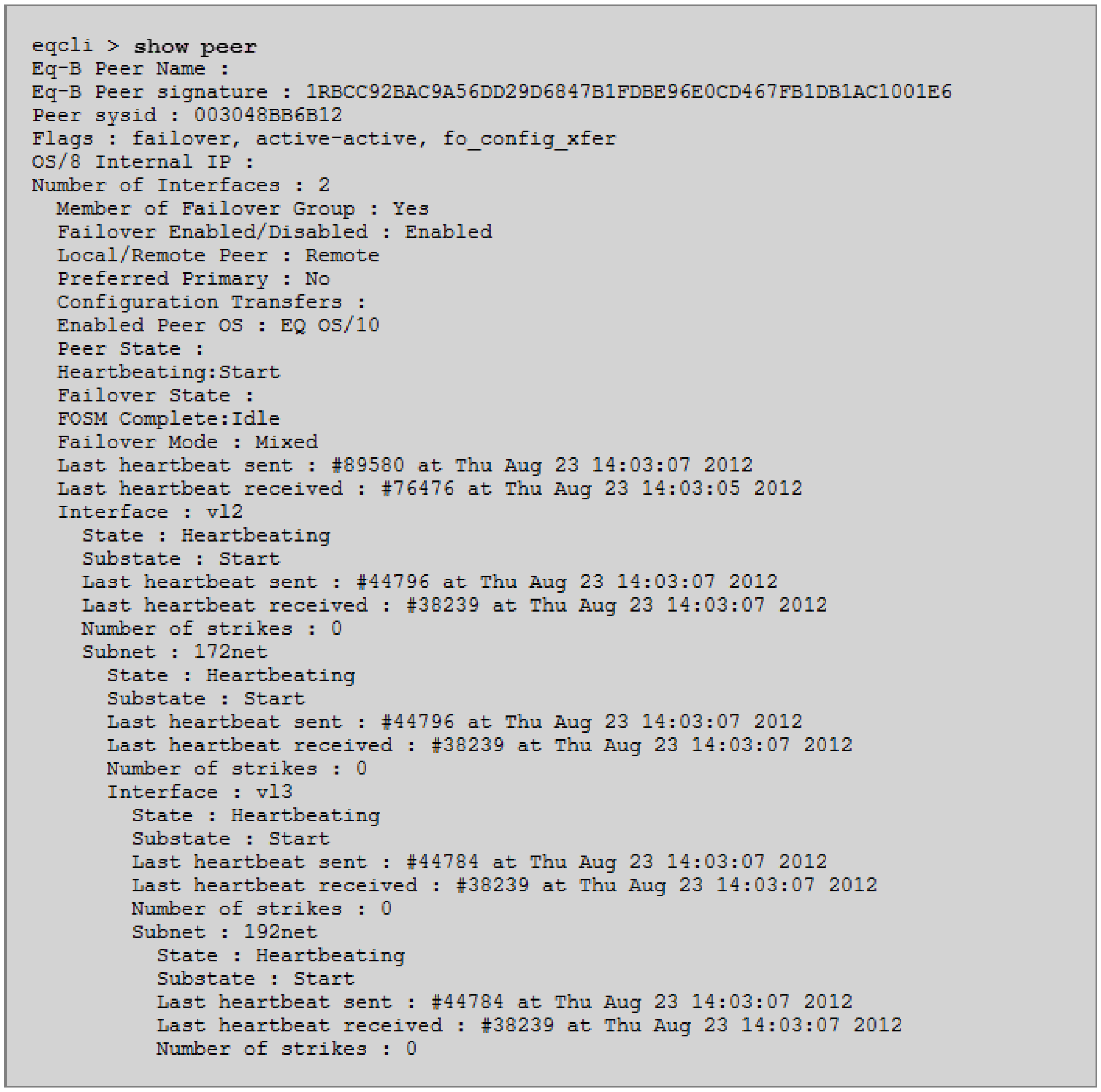
```
eqcli > show peer

Peer Name       Type    Flags                   F/O Mode        Error?
Eq-A (Local)    OS/10   F/O, A/A, P/P, xfr      Mixed           No
Eq-B (Remote)   OS/10   F/O, A/A, xfr           Mixed           No
Eq-C (Remote)   OS/10   F/O, A/A, xfr           Backup          No

Flags Key:
        F/O => failover
        A/A => active-active
        P/P => perferred_primary
        sfr => fo_config_xfer

eqcli >
```

For "N+1" failover:
1. Each peer should have the A/A (active-active) flag enabled
2. The modes displayed will be different for active-active, as explained below.

As cables are removed, re-attached, and systems rebooted, the `F/O Mode` displayed for each peer will move through the following failover mode variations:

Primary	The peer has instantiated all cluster IP addresses, and all subnet failover IP addresses. Heartbeating is working properly.
Mixed	The peer has instantiated some of the cluster IP addresses in the configuration and is available as a Backup for others. It has also instantiated all subnet failover IP addresses for the subnets required by the instantiated clusters.
Backup	The peer has not instantiated any clusters and is available as a Backup. Heartbeating is working properly.
Isolated	The peer appears to be up but we cannot heartbeat it. This usually occurs when a peer is rebooted and has not yet fully assumed a failover mode, or when there is a connectivity issue on a heartbeating subnet.
Unknown	The peer status is unknown. No heartbeats from this system have been received. This usually occurs when first configuring failover, before all peers have started heartbeating one another, or when rebooting a peer.
Standalone	The peer is not participating in failover. It will instantiate all cluster IP addresses and all subnet failover IP addresses that exist in the configuration file.

For detailed output regarding heartbeat status between this peer and other peers in the failover set of Equalizers, specify the name of a remote peer:



```
eqcli > show peer
Eq-B Peer Name :
Eq-B Peer signature : 1RBCC92BAC9A56DD29D6847B1FDBE96E0CD467FB1DB1AC1001E6
Peer sysid : 003048BB6B12
Flags : failover, active-active, fo_config_xfer
OS/8 Internal IP :
Number of Interfaces : 2
  Member of Failover Group : Yes
  Failover Enabled/Disabled : Enabled
  Local/Remote Peer : Remote
  Preferred Primary : No
  Configuration Transfers :
  Enabled Peer OS : EQ OS/10
  Peer State :
  Heartbeating:Start
  Failover State :
  FOSM Complete:Idle
  Failover Mode : Mixed
  Last heartbeat sent : #89580 at Thu Aug 23 14:03:07 2012
  Last heartbeat received : #76476 at Thu Aug 23 14:03:05 2012
  Interface : vl2
    State : Heartbeating
    Substate : Start
    Last heartbeat sent : #44796 at Thu Aug 23 14:03:07 2012
    Last heartbeat received : #38239 at Thu Aug 23 14:03:07 2012
    Number of strikes : 0
    Subnet : 172net
      State : Heartbeating
      Substate : Start
      Last heartbeat sent : #44796 at Thu Aug 23 14:03:07 2012
      Last heartbeat received : #38239 at Thu Aug 23 14:03:07 2012
      Number of strikes : 0
      Interface : vl3
        State : Heartbeating
        Substate : Start
        Last heartbeat sent : #44784 at Thu Aug 23 14:03:07 2012
        Last heartbeat received : #38239 at Thu Aug 23 14:03:07 2012
        Number of strikes : 0
        Subnet : 192net
          State : Heartbeating
          Substate : Start
          Last heartbeat sent : #44784 at Thu Aug 23 14:03:07 2012
          Last heartbeat received : #38239 at Thu Aug 23 14:03:07 2012
          Number of strikes : 0
```

Displaying Cluster Status

Specify the name of a cluster to the show cluster command to see if the cluster is currently instantiated on the Equalizer to which you are logged in. The first couple of lines in the output indicate the cluster status, as in this example:

```
eqcli > show cluster clB
This cluster has a problem:
Cluster is not active on this Equalizer

L7 Cluster Name          : clB
Protocol                 : http
IP Address               : 192.168.0.161
Port                     : 80
Preferred Peer           : Eq-B
VID                      : 3
...
```

The second line of output indicates that this cluster is not instantiated on this Equalizer; if this message does not appear, then the cluster should be instantiated on this Equalizer (assuming that there are no other issues, such as a cluster mis-configuration that is not related to failover).

Also shown in the output are the preferred peer and `VID` (VLAN ID) settings. Basic troubleshooting for failover includes verifying that all preferred peer and `VID` settings on clusters are correct.

Rebalancing

Rebalancing is usually done after a failover event occurs and all system have been returned to normal service. This instantiates each cluster (and its required objects, such as servers) on the peer set in the cluster's `preferred_peer` parameter. In the example configurations that follow, the clusters `clA` and `clB` will continue to run on Eq-A until you run this command on Eq-A:

```
eqcli > rebalance
```

This instantiates each cluster on the preferred peer set in its configuration. In this case, cluster clB will migrate from Eq-A to run on Eq-B (its preferred peer) instead. The clusters will continue to run on their preferred peers until a failover event occurs.

After rebalancing, the `F/O Mode` displayed for Eq-A and Eq-B should be `Mixed`. This indicates that it is acting as the primary system for some clusters and as backup for others.

Configuring N + 1 Failover with 3 Load Balancers (CLI)

In this configuration, three Equalizers (Eq-A, Eq-B, and Eq-C) cooperate to provide high availability. They do not need to be the same models. They are configured with:

- 2 VLAN subnets
- 2 clusters -- 1 preferred on each of Eq-A and Eq-B, no clusters on Eq-C
- 2 failover groups

1. Do the following on all three Load Balancers:

   a. Create all VLANs and subnets necessary for your configuration (see "Configuring VLANs" on page 304). For this example, we assume two VLANs: **vlan2** with two and **vlan3**) with one or two subnets each (**172net** and **192net**, respectively), and that these are cable-connected to the load balancer through separate front-panel ports. As with any failover configuration, the VLAN/subnet configuration on all peers must be exactly the same, except for object names and tagged/untagged port assignments.

   b. Set the Failover (or Virtual) IP address on each VLAN subnet, as in these examples:

```
eqcli > vlan vlan2 subnet 172net virt_addr 172.16.0.169
eqcli > vlan vlan3 subnet 192net virt_addr 192.168.0.169
```

   c. Set the command and heartbeat flags on the subnets. One subnet must have the command flag enabled, both subnets need the heartbeat flag since we want to fail over when there is a connectivity issue on any subnet:

```
eqcli > vlan vlan2 subnet 172net flags command,heartbeat
eqcli > vlan vlan3 subnet 192net flags heartbeat
```

   d. Change the system `hostname` so it is unique.This is useful when examining logs:

```
eqcli > hostname name
```

e. Set the timezone.The time zone setting is useful when examining logs. Enter:

```
eqcli > timezone?
```

Locate your timezone in the displayed list and press "q" to quit out of the list. Then, type in your timezone number and press <`Enter`>, as in this example for the "America/New York" time zone:

```
eqcli > timezone 161
```

f. If the load balancer can reach the Internet, add a name server so that NTP will work and time will be the same across all load balancers:

```
eqcli > name-server IP_address
```

Otherwise, set the time manually on all systems to the current time:

```
eqcli > date HHmmss
```

In the above command, **`HH`** is hours, mm is minutes, and **`ss`** is seconds. Seconds are optional.

g. Enable NTP. If you've defined at least one DNS server, you can configure the Network Time Protocol (NTP).by entering

```
eqcli > ntp enable
```

h. Change the name of the local peer so it’s easier to recognize, as in this example for Equalizer `Eq-A`:

```
eqcli > peer e<TAB> name Eq-A
```

Note that the <`TAB`> above means press the Tab key on your keyboard to auto-complete the local peer name. Since this unit currently has only one peer definition it fills it out with the local peer name.

2. After you complete Step 1 on *all three* load balancers, do the following on load balancer `Eq-A`:

   a. Create the clusters, servers, server pools, and server instances necessary for your configuration. For the purposes of this procedure, we created the following objects and non-default settings:

```
eqcli > server sv2 proto tcp ip 172.16.0.170 port 80
eqcli > srvpool sp01 policy adaptive
eqcli > srvpool sp01 si sv2 weight 100
eqcli > cluster clA proto http ip 172.16.0.160 port 80 srvpool sp01
eqcli > server sv3 proto tcp ip 192.168.0.24 port 80
eqcli > srvpool sp01 policy adaptive
eqcli > srvpool sp02 si sv3 weight 100
eqcli > cluster clB proto http ip 192.168.0.161 port 80 srvpool sp02
```

Note - In this procedure, we create all of the clusters, servers, and server pools on the preferred primary Equalizer, assign a preferred peer to each cluster, and then rebalance to move the clusters to their preferred peer Equalizers. You could also create your clusters on the other peers. If you do, be sure to specify a preferred peer for each cluster when you create them if you want them to be instantiated on that peer; otherwise, they will be instantiated on the peer that has the preferred primary flag enabled

   b. Update the flags for peer `Eq-A`:

```
eqcli > peer Eq-A flags failover,fo_config_xfer,preferred_
primary,active-active
```

   c. Create the peer definitions for the remote peers `Eq-B` and `Eq-C`:

```
eqcli > peer Eq-B signature signature
eqcli > peer Eq-C signature signature
```

Note - The signature for each remote peer can be displayed by logging into the CLI on that peer and executing "show peer name", where ***name*** is Eq-B or Eq-C.

   d. Set a preferred peer for each cluster:

```
eqcli > cluster clA preferred_peer Eq-A
eqcli > cluster clB preferred_peer Eq-B
```

3. Do the following on `Eq-B`:

   a. Update the flags for peer `Eq-B`:

```
eqcli > peer Eq-B flags failover,active-active. fo_config_xfer
```

   b. Create the peer definitions for the remote peers `Eq-A` and `Eq-C`:

```
eqcli > peer Eq-A signature signature
eqcli > peer Eq-C signature signature
```

Note - The signature for each remote peer can be displayed by logging into the CLI on that peer and executing "**`show peer name`**", where name is `Eq-A` or `Eq-C`.

4. Do the following on `Eq-C`:

   a. Update the flags for peer `Eq-C`:

```
eqcli > peer Eq-C flags failover,active-active, fo_config_xfer
```

   b. Create the peer definitions for the remote peers `Eq-A` and `Eq-B`:

```
eqcli > peer Eq-A signature signature
eqcli > peer Eq-B signature signature
```

Note - The signature for each remote peer can be displayed by logging into the CLI on that peer and executing "**`show peer name`**", where name is `Eq-A` or `Eq-B`.

5. Verify that the peer status is correct:
   a. On `Eq-A` the peer status should now look like this:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: XXXX
------------------------------------
Peer Name     Type    Flags              F/O Mode     Message(s)?
Eq-A (Local) OS/10   F/O, A/A, P/P, xfr Primary      No
Eq-B (Remote)OS/10   F/O, A/A, xfr      Backup       No
Eq-C (Remote)OS/10   F/O, A/A, xfr      Backup       No
```

   b. On `Eq-B`, the peer status should now look like this:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: XXXX
------------------------------------
Peer Name     Type    Flags              F/O Mode     Message(s)?
Eq-B (Local) OS/10   F/O, A/A, xfr      Backup       No
Eq-A (Remote)OS/10   F/O, A/A, P/P, xfr Primary      No
Eq-C (Remote)OS/10   F/O, A/A, xfr      Backup       No
```

   c. On `Eq-C`, the peer status should now look like this:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: XXXX
------------------------------------
Peer Name     Type    Flags              F/O Mode     Message(s)?
Eq-C (Local) OS/10   F/O, A/A, xfr      Backup       No
Eq-B (Remote)OS/10   F/O, A/A, xfr      Backup       No
Eq-A (Remote)OS/10   F/O, A/A, P/P, xfr Primary      No
```

6. Show the fo group details and status as follows:
   a. For `fo_group1`:

```
eqcli > show fogrp fo_group1
F/O Group fo_group1:
ID                        = 1
Preferred Peer            = Eq-A
Primary Peer              = Eq-A
F/O Mode                  = Primary
Subnet Members (num = 2):
   vlan2:172net
   vlan3:192net
Cluster Members (num = 2):
   c1A
   clB
Server Members (num = 2):
   sv2
   sv3
eqcli >
```

   b. For `fo_group2`:

```
eqcli > show fogrp fo_group2
F/O Group fo_group1:
ID                        = 1
Preferred Peer            = Eq-B
Primary Peer              = Eq-A
F/O Mode                  = Backup
Subnet Members (num = 2):
   vlan2:172net
   vlan3:192net
Cluster Members (num = 2):
   c1A
   clB
Server Members (num = 2):
   sv2
   sv3
eqcli >
```

After the above procedure is completed, the object configuration should get synchronized over to Eq-B and Eq-C. All Equalizerobjects will be visible in the CLI and GUI of all peers. The two clusters will continue to run on Eq-A until they are "rebalanced" (Refer to "Rebalancing" on page 742).

Configuring N + 1 Failover with 4 Load Balancers (CLI)

In this configuration, four Equalizers (Eq-A, Eq-B, Eq-C, and Eq-D) cooperate to provide high availability. They do not need to be the same models. They are configured with:

- 3 VLAN subnets
- 3 clusters -- 1 preferred on each of Eq-A, Eq-B, and Eq-C; no clusters on Eq-D
- 3 failover groups

1. Do the following on all four Load Balancers:
   a. Create all VLANs and subnets necessary for your configuration (see "Configuring VLANs" on page 304). For this example, we assume two VLANs (**vlan2** with two subnets and **vlan3** with one. These are cabled to Equalizer through separate front-panel ports. As with any failover configuration, the VLAN/subnet configuration on all peers must be exactly the same, except for object names and tagged/untagged port assignments.
   b. Set the Failover (or Virtual) IP address on each vlan subnet, as in these examples:

```
eqcli > vlan vlan2 subnet 172net-1 virt_addr 172.16.0.169
eqcli > vlan vlan2 subnet 172net-2 virt_addr 172.16.1.169
eqcli > vlan vlan3 subnet 192net virt_addr 192.168.0.169
```

   c. Set the command and heartbeat flags on the subnets. One subnet must have the command flag enabled, all subnets need the heartbeat flag since we want to fail over when there is a connectivity issue on any subnet:

```
eqcli > vlan vlan2 subnet 172net-1 flags command,heartbeat
eqcli > vlan vlan2 subnet 172net-2 flags heartbeat
eqcli > vlan vlan3 subnet 192net flags heartbeat
```

   d. Change the system `hostname` so it is unique. This is useful when examining logs:

```
eqcli > hostname name
```

e. Set the timezone. Enter:

```
eqcli > timezone?
```

Locate your timezone in the displayed list and press "q" to quit out of the list. Then, type in your timezone number and press `<Enter>`, as in this example for the "America/New York" time zone:

```
eqcli > timezone 161
```

f. If Equalizer can reach the Internet, add a name server so that NTP will work and time will be the same across all Equalizer:

```
eqcli > name-server IP_address
```

Otherwise, set the time manually on all systems to the current time:

```
eqcli > date HHmmss
```

In the above command, **HH** is hours, **mm** is minutes, and **ss** is seconds. Seconds are optional.

g. Enable NTP. If you've defined at least one DNS server, you can configure the Network Time Protocol (NTP).by entering

```
eqcli > ntp enable
```

h. Change the name of the local peer so it’s easier to recognize, as in this example for Equalizer `Eq-A`:

```
eqcli > peer e<TAB> name Eq-A
```

Note - Note that the `<TAB>` above means press the Tab key on your keyboard to auto-complete the local peer name. Since this unit currently has only one peer definition it fills it out with the local peer name.

2. After you complete Step 1 on all *three* Equalizers, do the following on Equalizer `Eq-A`:

   a. Create the clusters, servers, server pools, and server instances necessary for your configuration. For the purposes of this procedure, we created the following objects and non-default settings:

```
eqcli > server sv2 proto tcp ip 172.16.0.170 port 80
eqcli > srvpool sp01 policy adaptive
eqcli > srvpool sp01 si sv2 weight 100
eqcli > cluster clA proto http ip 172.16.0.160 port 80 srvpool sp01
eqcli > server sv3 proto tcp ip 172.16.1.170 port 80
eqcli > srvpool sp02 policy adaptive
eqcli > srvpool sp02 si sv3 weight 100
eqcli > cluster clB proto http ip 172.16.1.160 port 80 srvpool sp02
eqcli > server sv4 proto tcp ip 192.168.0.24 port 80
eqcli > srvpool sp03 policy adaptive
eqcli > srvpool sp03 si sv4 weight 100
eqcli > cluster clC proto http ip 192.168.0.161 port 80 srvpool sp03
```

Note - In this procedure, we create all of the clusters, servers, and server pools on the preferred primary Equalizer, assign a preferred peer to each cluster, and then rebalance to move the clusters to their preferred peer Equalizers. You could also create your clusters on the other peers. If you do, be sure to specify a preferred peer for each cluster when you create them if you want them to be instantiated on that peer; otherwise, they will be instantiated on the peer that has the preferred primary flag enabled.

   b. Update the flags for peer `Eq-A`:

```
eqcli > peer Eq-A flags failover,fo_config_xfer,preferred_
primary,active-active
```

   c. Create the peer definitions for the remote peers `Eq-B` and `Eq-C`:

```
eqcli > peer Eq-B signature signature
eqcli > peer Eq-C signature signature
eqcli > peer Eq-D signature signature
```

Note - The signature for each remote peer can be displayed by logging into the CLI on that peer and executing "**`show peer name`**", where name is Eq-B, Eq-C, or Eq-D.

d. Set a preferred peer for each cluster:

```
eqcli > cluster clA preferred_peer Eq-A
eqcli > cluster clB preferred_peer Eq-B
eqcli > cluster clC preferred_peer Eq-C
```

3. Do the following on `Eq-B`:

a. Update the flags for peer `Eq-B`:

```
eqcli > peer Eq-B flags failover,active-active,fo_config_xfer
```

b. Create the peer definitions for the remote peers `Eq-A`, `Eq-C`, and `Eq-D`:

```
eqcli > peer Eq-A signature signature flags failover,fo_config_
xfer,preferred_primary
 eqcli > peer Eq-C signature signature flags failover
eqcli > peer Eq-D signature signature flags failover
```

Note - The signature for each remote peer can be displayed by logging into the CLI on that peer and executing "`show peer name`", where name is Eq-A, Eq-C, and Eq-D.

4. Do the following on `Eq-C`:

a. Update the flags for peer `Eq-C`:

```
eqcli > peer Eq-C flags failover,active-active,fo_config_xfer
```

b. Create the peer definitions for the remote peers E`q-A`, `Eq-B`, and `Eq-D`:

```
eqcli > peer Eq-A signature signature
eqcli > peer Eq-B signature signature
eqcli > peer Eq-D signature signature
```

Note - The signature for each remote peer can be displayed by logging into the CLI on that peer and executing **`show peer`** ***`name`***, where ***`name`*** is Eq-A, Eq-B, or Eq-D.

5. Do the following on `Eq-D`:

   a. Update the flags for peer `Eq-D`:

```
eqcli > peer Eq-D flags failover,active-active, fo_config_xfer
```

   b. Create the peer definitions for the remote peers `Eq-A`, `Eq-B`, and `Eq-C`:

```
eqcli > peer Eq-A signature signature flags failover,fo_config_
xfer,preferred_primary
eqcli > peer Eq-B signature signature flags failover
eqcli > peer Eq-C signature signature flags failover
```

Note - The signature for each remote peer can be displayed by logging into the CLI on that peer and executing "**show peer name**", where name is **Eq-A**, **Eq-B**, or **Eq-C**.

6. Verify that the peer status is correct:

   a. On `Eq-A`, the peer status should now look like this:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: XXXX
------------------------------------
Peer Name    Type       Flags               F/O Mode  Message(s)
Eq-A (Local) OS/10      F/O, A/A, P/P, xfr  Primary   No
Eq-B (Remote)OS/10      F/O, A/A, xfr       Backup    No
Eq-C (Remote)OS/10      F/O, A/A, xfr       Backup    No
Eq-D (Remote)OS/10      F/O, A/A, xfr       Backup    No
```

   b. On `Eq-B`, the peer status should now look like this:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: XXXX
------------------------------------
Peer Name    Type       Flags               F/O Mode  Message(s)
Eq-B (Local) OS/10      F/O, A/A, xfr       Backup    No
Eq-A (Remote)OS/10      F/O, A/A, P/P, xfr  Primary   No
Eq-C (Remote)OS/10      F/O, A/A, xfr       Backup    No
Eq-D (Remote)OS/10      F/O, A/A, xfr       Backup    No
```

c. On `Eq-C`, the peer status should now look like this:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: XXXX
------------------------------------
Peer Name    Type       Flags              F/O Mode    Message(s)
Eq-C (Local) OS/10      F/O, A/A, xfr      Backup      No
Eq-A (Remote)OS/10      F/O, A/A, P/P, xfr Primary     No
Eq-B (Remote)OS/10      F/O, A/A, xfr      Backup      No
Eq-D (Remote)OS/10      F/O, A/A, xfr      Backup      No
```

d. On `Eq-D`, the peer status should now look like this:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: XXXX
------------------------------------
Peer Name    Type       Flags              F/O Mode    Message(s)
Eq-D (Local) OS/10      F/O, A/A, xfr      Backup      No
Eq-A (Remote)OS/10      F/O, A/A, P/P, xfr Primary     No
Eq-B (Remote)OS/10      F/O, A/A, xfr      Backup      No
Eq-C (Remote)OS/10      F/O, A/A, xfr      Backup      No
```

If all peers sharing several failover groups are rebooted or powered on in a sequential fashion (first reboot `Eq-A`, then `Eq-B` etc.), the expected behavior is that one unit may become Primary for all failover groups, depending upon the sequence in which the systems become active on the network. If this occurs, running the "rebalance" command will re-distribute the failover groups to their preferred primary Equalizers.

7. Show the fo group details and status as follows:
   a. For `fo_group1`:

```
eqcli > show fogrp fo_group1
F/O Group fo_group1:
ID                        = 1
Preferred Peer            = Eq-A
Primary Peer              = Eq-A
F/O Mode                  = Primary
Subnet Members (num = 3):
   vlan2:172net-1
   vlan2:172net-2
   vlan3:192net
Cluster Members (num = 3):
   clA
   clB
   clC
Server Members (num = 3):
   sv2
   sv3
   sv4
eqcli >
```

   b. For `fo_group2`:

```
eqcli > show fogrp fo_group2
F/O Group fo_group2:
ID                        = 1
Preferred Peer            = Eq-B
Primary Peer              = Eq-A
F/O Mode                  = Backup
Subnet Members (num = 3):
   vlan2:172net-1
   vlan2:172net-2
   vlan3:192net
Cluster Members (num = 3):
   clA
   clB
   clC
Server Members (num = 3):
   sv2
   sv3
   sv4
eqcli >
```

c. For `fo_group3`:

```
eqcli > show fogrp fo_group3
F/O Group fo_group3:
ID                     = 1
Preferred Peer         = Eq-C
Primary Peer           = Eq-A
F/O Mode               = Backup
Subnet Members (num = 3):
   vlan2:172net-1
   vlan2:172net-2
   vlan3:192net
Cluster Members (num = 3):
   clA
   clB
   clC
Server Members (num = 3):
   sv2
   sv3
   sv4
eqcli >
```

After the above procedure is completed, the configuration on Eq-A should get copied over to Eq-B, Eq-C, and Eq-D. All Equalizer objects will be visible in the CLI and GUI of all peers. All clusters will continue to run on Eq-A until they are "rebalanced" or a failover occurs. (Refer to "Rebalancing" on page 742).

Configuring N + 0 Failover with 4 Load Balancers (CLI)

In this configuration, four Equalizers (Eq-A, Eq-B, Eq-C, and Eq-D) cooperate to provide high availability. They do not need to be the same models. They are configured with:

- 4 VLAN subnets
- 4 clusters -- 1 preferred on each of Eq-A, Eq-B, Eq-C, and Eq-D
- 4 failover groups

1. Do the following on all four Load Balancers:

    a. Create all VLANs and subnets necessary for your configuration (see "Configuring VLANs" on page 304). For this example, we assume two VLANs (**vlan2** with two subnets and **vlan3** with one. These are cabled to Equalizer through separate front-panel ports. As with any failover configuration, the VLAN/subnet configuration on all peers must be exactly the same, except for object names and tagged/untagged port assignments.

    b. Set the Failover (or Virtual) IP address on each vlan subnet, as in these examples:

```
eqcli > vlan vlan2 subnet 172net-1 virt_addr 172.16.0.169
eqcli > vlan vlan2 subnet 172net-2 virt_addr 172.16.1.169
eqcli > vlan vlan3 subnet 192net-1 virt_addr 192.168.0.169
eqcli > vlan vlan3 subnet 192net-2 virt_addr 192.168.1.169
```

    c. Set the command and heartbeat flags on the subnets. One subnet must have the command flag enabled, all subnets need the heartbeat flag since we want to fail over when there is a connectivity issue on any subnet:

```
eqcli >vlan vlan2 subnet 172net-1 flags command,heartbeat
eqcli >vlan vlan2 subnet 172net-2 flags heartbeat
eqcli >vlan vlan3 subnet 192net-1 flags heartbeat
eqcli >vlan vlan3 subnet 192net-2 flags heartbeat
```

    d. Change the system `hostname` so it is unique. This is useful when examining logs:

```
eqcli > hostname name
```

e. Set the timezone. Enter:

```
eqcli > timezone?
```

Locate your timezone in the displayed list and press "q" to quit out of the list. Then, type in your timezone number and press <`Enter`>, as in this example for the "America/New York" time zone:

```
eqcli > timezone 161
```

f. If Equalizer can reach the Internet, add a name server so that NTP will work and time will be the same across all Equalizers:

```
eqcli > name-server IP_address
```

Otherwise, set the time manually on all systems to the current time:

```
eqcli > date HHmmss
```

In the above command, ***HH*** is hours, ***mm*** is minutes, and ***ss*** is seconds. Seconds are optional.

g. Enable NTP. If you've defined at least one DNS server, you can configure the Network Time Protocol (NTP).by entering

```
eqcli > ntp enable
```

h. Change the name of the local peer so it’s easier to recognize, as in this example for Equalizer `Eq-A`:

```
eqcli > peer <TAB> name Eq-A
```

Note - The <`TAB`> above means press the Tab key on your keyboard to auto-complete the local peer name. Since this unit currently has only one peer definition it fills it out with the local peer name.

2. After you complete Step 1 on all three Equalizers, do the following on Equalizer `Eq-A`:

   a. Create the clusters, servers, server pools, and server instances necessary for your configuration. For the purposes of this procedure, we created the following objects and non-default settings:

```
eqcli > server sv2 proto tcp ip 172.16.0.170 port 80
eqcli > srvpool sp01 policy adaptive
eqcli > srvpool sp01 si sv2 weight 100
eqcli > cluster clA proto http ip 172.16.0.161 port 80 srvpool sp01
eqcli > server sv3 proto tcp ip 172.16.1.170 port 80
eqcli > srvpool sp02 policy adaptive
eqcli > srvpool sp02 si sv3 weight 100
eqcli > cluster clB proto http ip 172.16.1.161 port 80 srvpool sp02
eqcli > server sv4 proto tcp ip 192.168.0.24 port 80
eqcli > srvpool sp03 policy adaptive
eqcli > srvpool sp03 si sv4 weight 100
eqcli > cluster clC proto http ip 192.168.0.161 port 80 srvpool sp03
eqcli > server sv5 proto tcp ip 192.168.1.24 port 80
eqcli > srvpool sp04 policy adaptive
eqcli > srvpool sp04 si sv5 weight 100
eqcli > cluster clC proto http ip 192.168.1.161 port 80 srvpool sp04
```

Note - In this procedure, we create all of the clusters, servers, and server pools on the preferred primary Equalizer, assign a preferred peer to each cluster, and then rebalance to move the clusters to their preferred peer Equalizers.

You could also create your clusters on the other peers. If you do, be sure to specify a preferred peer for each cluster when you create them if you want them to be instantiated on that peer; otherwise, they will be instantiated on the peer that has the preferred primary flag enabled.

   b. Update the flags for peer `Eq-A`:

```
eqcli > peer Eq-A flags failover,fo_config_xfer,preferred_
primary,active-active
```

c. Create the peer definitions for the remote peers `Eq-B` and `Eq-C`:

```
eqcli > peer Eq-B signature signature
eqcli > peer Eq-C signature signature
eqcli > peer Eq-D signature signature
```

Note - The signature for each remote peer can be displayed by logging into the CLI on that peer and executing **`show peer name`**, where ***`name`*** is Eq-B, Eq-C, or Eq-D.

d. Set a preferred peer for each cluster:

```
eqcli > cluster clA preferred_peer Eq-A
eqcli > cluster clB preferred_peer Eq-B
eqcli > cluster clC preferred_peer Eq-C
eqcli > cluster clD preferred_peer Eq-D
```

3. Do the following on `Eq-B`:

a. Update the flags for peer Eq-B:

```
eqcli > peer Eq-B flags failover,active-active,fo_config_xfer
```

b. Create the peer definitions for the remote peers `Eq-A`, `Eq-C`, and `Eq-D`:

```
eqcli > peer Eq-A signature signature
eqcli > peer Eq-C signature signature
eqcli > peer Eq-D signature signature
```

Note - The signature for each remote peer can be displayed by logging into the CLI on that peer and executing "**`show peer name`**", where ***`name`*** is **`Eq-A`**, **`Eq-C`**, and **`Eq-D`**.

4. Do the following on `Eq-C`:

   a. Update the flags for peer `Eq-C`:

   ```
   eqcli > peer Eq-C flags failover,active-active,fo_config_xfer
   ```

   b. Create the peer definitions for the remote peers `Eq-A`, `Eq-B`, and `Eq-D`:

   ```
   eqcli > peer Eq-A signature signature
   eqcli > peer Eq-B signature signature
   eqcli > peer Eq-D signature signature
   ```

Note - The signature for each remote peer can be displayed by logging into the CLI on that peer and executing "**show peer name**", where ***name*** is **Eq-A**, **Eq-B**, or **Eq-D**.

5. Do the following on `Eq-D`:

   a. Update the flags for peer `Eq-D`:

   ```
   eqcli > peer Eq-D flags failover,active-active,fo_config_xfer
   ```

   b. Create the peer definitions for the remote peers `Eq-A`, `Eq-B`, and `Eq-C`:

   ```
   eqcli > peer Eq-A signature signature
   eqcli > peer Eq-B signature signature
   eqcli > peer Eq-C signature signature
   ```

Note - The signature for each remote peer can be displayed by logging into the CLI on that peer and executing "**show peer name**", where ***name*** is **Eq-A**, **Eq-B**, or **Eq-C**.

6. Verify that the peer status is correct:

   a. On `Eq-A`, the peer status should now look like this:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: XXXX
------------------------------------
Peer Name     Type      Flags           F/O mode        Message(s)
Eq-A (Local)  OS/10     F/O,A/A,P/P,xfr Primary         No
Eq-B (Remote) OS/10     F/O,A/A,xfr     Backup          No
Eq-C (Remote) OS/10     F/O,A/A,xfr     Backup          No
Eq-D (Remote) OS/10     F/O,A/A,xfr     Backup          No
```

   b. On `Eq-B`, the peer status should now look like this:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: XXXX
------------------------------------
Peer Name     Type      Flags           F/O mode      Message(s)
Eq-B (Local)  OS/10     F/O,A/A,xfr     Backup        No
Eq-A (Remote) OS/10     F/O,A/A,P/P,xfr Primary       No
Eq-C (Remote) OS/10     F/O,A/A,xfr     Backup        No
Eq-D (Remote) OS/10     F/O,A/A,xfr     Backup        No
```

   c. On `Eq-C`, the peer status should now look like this:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: XXXX
------------------------------------
Peer Name     Type      Flags           F/O mode      Message(s)
Eq-C (Local)  OS/10     F/O,A/A,xfr     Backup        No
Eq-A (Remote) OS/10     F/O,A/A,P/P,xfr Primary       No
Eq-B (Remote) OS/10     F/O,A/A,xfr     Backup        No
Eq-D (Remote) OS/10     F/O,A/A,xfr     Backup        No
```

d. On `Eq-D`, the peer status should now look like this:

```
eqcli > show peer
------------------------------------
Configuration Sequence Number: XXXX
------------------------------------
Peer Name      Type      Flags            F/O mode     Message(s)
Eq-D (Local)  OS/10     F/O,A/A,xfr      Backup       No
Eq-A (Remote) OS/10     F/O,A/A,P/P,xfr Primary      No
Eq-B (Remote) OS/10     F/O,A/A,xfr      Backup       No
Eq-D (Remote) OS/10     F/O,A/A,xfr      Backup       No
```

7. Show the fo group details and status as follows:
   a. For `fo_group1`:

```
eqcli > show fogrp fo_group1
F/O Group fo_group1:
ID                       = 1
Preferred Peer           = Eq-A
Primary Peer             = Eq-A
F/O Mode                 = Primary
Subnet Members (num = 4):
   vlan2:172net-1
   vlan2:172net-2
   vlan3:192net-1
   vlan3:192net-2
Cluster Members (num = 4):
   clA
   clB
   clC
   clD
Server Members (num = 4):
   sv2
   sv3
   sv4
   sv5
eqcli >
```

b. For `fo_group2`:

```
eqcli > show fogrp fo_group2
F/O Group fo_group2:
ID                       = 1
Preferred Peer           = Eq-B
Primary Peer             = Eq-A
F/O Mode                 = Backup
Subnet Members (num = 4):
   vlan2:172net-1
   vlan2:172net-2
   vlan3:192net-1
   vlan3:192net-2
Cluster Members (num = 4):
   clA
   clB
   clC
   clD
Server Members (num = 4):
   sv2
   sv3
   sv4
   sv5
eqcli >
```

c. For `fo_group3`:

```
eqcli > show fogrp fo_group3
F/O Group fo_group3:
ID                       = 1
Preferred Peer           = Eq-C
Primary Peer             = Eq-A
F/O Mode                 = Backup
Subnet Members (num = 4):
   vlan2:172net-1
   vlan2:172net-2
   vlan3:192net-1
   vlan3:192net-2
Cluster Members (num = 4):
   clA
   clB
   clC
   clD
Server Members (num = 4):
   sv2
   sv3
   sv4
   sv5
eqcli >
```

d. For `fo_group4`:

```
eqcli > show fogrp fo_group4
F/O Group fo_group4:
ID                    = 1
Preferred Peer        = Eq-D
Primary Peer          = Eq-A
F/O Mode              = Backup
Subnet Members (num = 4):
   vlan2:172net-1
   vlan2:172net-2
   vlan3:192net-1
   vlan3:192net-2
Cluster Members (num = 4):
   clA
   clB
   clC
   clD
Server Members (num = 4):
   sv2
   sv3
   sv4
   sv5
eqcli >
```

After the above procedure is completed, the object configuration should get synchronized over to Eq-B, Eq-C, and Eq-D. All Equalizerobjects will be visible in the CLI and GUI of all peers. All clusters will continue to run on Eq-A until they are "rebalanced" or a failover event occurs. (Refer to "Rebalancing" on page 742).

Chapter 22

Configuring Server Connections

Sections within this chapter include:

HTTP Multiplexing

HTTP multiplexing is the re-use of established server connections for multiple clients connections. The best way to understand this feature is to compare non-multiplexing behavior to multiplexing behavior. When HTTP multiplexing is disabled (the default on Equalizer), each client connection requires a new connection between Equalizer and a server.

What this means is that the servers behind Equalizer have to allocate a significant amount of resources to establishing and tearing down TCP connections -- resources that could otherwise be used by the applications running on the servers.

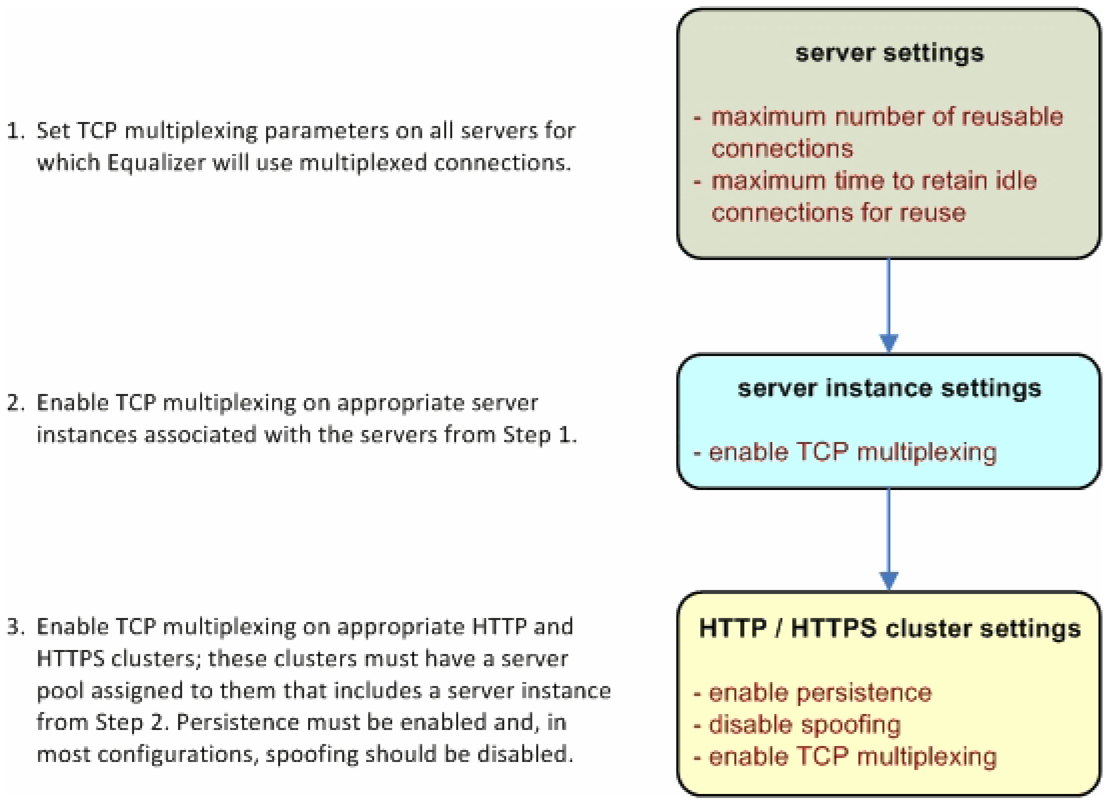
When HTTP multiplexing is enabled, an established server connection is left open for a period of time to see if any new client connections are load balanced to the same server. If so, this connection is used to forward the new client request to the server.

This allows Equalizer to service multiple client requests without all the overhead associated with establishing a new server connection for every request - and results in better performance on the client and server as well:

- the client does not have to wait for Equalizer to establish a server connection before sending the request to a server
- the server does not have to incur the overhead of establishing a new connection with Equalizer

Enabling HTTP Multiplexing

On Equalizer, TCP multiplexing can be enabled for HTTP and HTTPS clusters only and is disabled by default. The figure below describes the general process to follow when enabling TCP multiplexing for the first time.


1. Set TCP multiplexing parameters on all servers for which Equalizer will use multiplexed connections.
server settings
- maximum number of reusable connections
- maximum time to retain idle connections for reuse
2. Enable TCP multiplexing on appropriate server instances associated with the servers from Step 1.
server instance settings
- enable TCP multiplexing
3. Enable TCP multiplexing on appropriate HTTP and HTTPS clusters; these clusters must have a server pool assigned to them that includes a server instance from Step 2. Persistence must be enabled and, in most configurations, spoofing should be disabled.
HTTP / HTTPS cluster settings
- enable persistence
- disable spoofing
- enable TCP multiplexing

After TCP multiplexing is enabled as above, it can be selectively disabled on clusters and server instances without modifying the TCP multiplexing parameters set on the server.

Refer to "Modifying a Layer 7 HTTP or HTTPS Cluster" on page 362 or "Cluster and Match Rule Commands" on page 171 (on the CLI) for details.

Disabling "spoof" for HTTP Multiplexing

In the most common configurations, where many clients with unique IP addresses connect to the cluster, it makes sense to disable the **spoof** option when enabling TCP multiplexing, so that server connections can be re-used for any client request.

This is because the **spoof** option causes Equalizer to use the client IP address as the source address in all packets sent to servers (disabling Source Network Address Translation or SNAT). While this itself is not a problem, it means that server connections can only be re-used by client connections from the *same* client IP. This effectively disables much of the benefit of using TCP multiplexing. If the application running on the servers behind an Equalizer cluster requires the real client IP address in incoming requests (that is, **spoof** enabled), then in most configurations we recommend disabling TCP multiplexing.

In some cases, when it is known that most or all client connections will come from a relatively short list of IP addresses, **spoof** can be enabled with TCP multiplexing to improve performance. Examples include configurations where public client connections come from an HTTP or HTTPS proxy that uses a restricted set of IP address, or an internal corporate network that uses NAT.

Refer to "Modifying a Layer 7 HTTP or HTTPS Cluster" on page 362 or "Cluster and Match Rule Commands" on page 171 (on the CLI) for details.

Server Options for HTTP Multiplexing

Once a server sends a complete response to a client request, instead of closing the server connection, Equalizer keeps the connection open and places a record for the connection into a pool of connections available for re-use. The connection will be re-used by Equalizer when another client request is load balanced to the same server.

The reusable connection pool record for a server connection is only removed when either the server closes the connection, or the **Reused Connection Timeout** expires (see below).

The following server parameters for HTTP multiplexing control the size of the connection re-use pool.

- **Maximum Reused Connections (**An integer specifying the maximum number of reusable connection entries for this server allowed in the reusable server pool. The default is **0**, which means that there is no limit on the number of reusable connection pool entries.

  If you have HTTP multiplexing enabled and CPU or memory usage on Equalizer is significant, you can use this parameter to limit the size of the reusable connection pool -- which in turn limits the amount of memory and CPU resources used to manage HTTP multiplexing.

  You may also want to limit the number of reused connections to a server if the server is experiencing resource issues related to maintaining open connections with Equalizer.
- **Reused Connection Timeout (**The number of seconds after which a connection record for an idle connection in the reusable connection pool is removed, and the connection closed. The default value is **0** seconds, which means that records in the reusable connections pool never expire.

Direct Server Return (DSR)

In a typical load balancing scenario, server responses to client requests are routed through Equalizer on their way back to the client. Equalizer examines the headers of each response and may insert a cookie, before sending the server response on to the client.

In a Direct Server Return (DSR) configuration, the server receiving a client request responds directly to the client IP, bypassing Equalizer. Because Equalizer only processes incoming requests, cluster performance is dramatically improved when using DSR in high bandwidth applications, especially those that deliver a significant amount of streaming content. In such applications, it is not necessary for Equalizer to receive and examine the server's responses: the client makes a request and the server simply streams a large amount of data to the client.

DSR is supported on Layer 4 TCP and UDP clusters only, and is not supported for FTP clusters (Layer 4 TCP clusters with a start port of 21).

DSR configurations are often configured on a single VLAN or subnet, where the cluster IP and the server IPs are all on the internal interface. Refer to "Configuring Direct Server Return" on page 424 for details.

DSR can also be used in multiple VLAN configurations, although this is less common. Cluster IP addresses are on one VLAN/subnet, while server IP addresses are on another VLAN/subnet.

In any DSR configuration, note that the incoming client traffic is assumed to originate on the other side of the gateway device for the subnets on which Equalizer and the servers reside. The servers will usually have their default gateway set to something other than Equalizer so that they can respond directly to client requests.

In DSR configurations where a client device resides on the same side of the gateway as the DSR servers, there is the possibility that the servers will receive the ARP (Address Resolution Protocol) request for the virtual cluster IP address. Since the cluster IP address is configured on the loopback interface of each server (See "Configuring Direct Server Return" on page 424), one or more may respond to the ARP request. The client, and possibly even the gateway, will then route requests for the cluster IP to servers directly without going through Equalizer. If this occurs, you need to reconfigure the servers so that they do not respond to ARP requests for the cluster IP addresses configured on the loopback interface. The procedure to follow to do this is specific to the operating system running on the servers, so please consult the documentation for your server operating system.

Configuring a Cluster for Direct Server Return

The cluster **dsr** and **spoof** flags must be enabled for direct server return connections. In addition, the cluster **idle timeout** parameter should be set as described in the table below:

dsr	Enables Direct Server Return. All requests to this cluster IP will be forwarded to the server with the client IP as the source IP, and the cluster IP as the destination IP. The loopback interface of the server must be configured with the cluster IP to receive the requests.
spoof	- spoof causes Equalizer to spoof the client IP address when Equalizer routes a request to a server in a virtual cluster; that is, the IP address of the client is sent to the server, not the IP address of the Equalizer. This flag must be enabled for DSR.
idle timeout	The time in seconds before reclaiming idle Layer 4 connection records. Applies to Layer 4 TCP clusters only. For DSR, **idle timeout** must be set to a non-zero value, or Equalizer will never reclaim connection records for connections terminated by the server. The cluster's **idle timeout** should be set to the longest period within your application that you would like Equalizer to wait for consecutive messages from the client (since the Equalizer does not see server packets on DSR connections). For example, if the longest expected server response time and the longest expected delay between client responses on active connections are both 60 seconds, then set the **idle timeout** to 120 seconds.

The general procedure for configuring DSR on a new or existing cluster is as follows:

1. Enable the **dsr** and **spoof** flags on the cluster.
2. If the cluster is a Layer 4 TCP cluster and the **idle timeout** parameter is set to **0**, increase it as described in the table above.
3. Perform the procedure on each server in the server pool associated with the cluster.

Configuring Servers for Direct Server Return

1. Server configuration for DSR involves these basic steps:
2. Add a *loopback* network interface on the server.
3. Configure the loopback interface with the IP address and port of the DSR cluster.
4. Edit the configuration of the application on the server to listen for connections on the cluster IP and port. (An HTTP server, for example, returns a Bad Hostname error to the client if there is an IP mismatch.)
5. Check the routing on your network to ensure that traffic is being routed as expected. For example, Equalizer is usually *not* going to be used as the default gateway on your servers, since we want the servers to respond to clients directly. In most DSR configurations, the default gateway used on servers is the gateway most appropriate for reaching the client network. If routes are also needed through Equalizer, they should be configured through static routes on the servers.

See the Related Topics below for examples of configuring the loopback adapter and an HTTP server on Windows and Linux platforms for DSR.

Configuring Windows Server 2003 and IIS for DSR

The basic procedure below also applies to Windows XP and other versions of Windows.

1. Open **Start > Control Panel** and double-click **Network Connections.**
2. Select **View > Tiles.** If a **Microsoft Loopback Adapter** is already listed, proceed to the next step. Otherwise, to install the loopback interface as follows:

   a. Open **Start > Control Panel > Add Hardware**, and then click **Next.**

   b. Click **Yes, I have already connected the hardware**, and then click Next.

   c. At the bottom of the list, click **Add a new hardware device**, and then click **Next.**

   d. Click **Install the hardware that I manually select from a list**, and then click **Next.**

   e. Click **Network adapters**, and then click **Next.**

   f. In the **Manufacturer** box, click **Microsoft.**

   g. In the **Network Adapter** box, click **Microsoft Loopback Adapter**, and then click **Next.**

   h. Click **Finish.**

3. To configure the loopback interface for DSR:

   a. In **Network Connections**, right click on the **Microsoft Loopback Adapter** and select **Properties.**

   b. In the **General** tab, double-click on **Internet Protocol (TCP/IP)** in the scroll box.

   c. Select **Use the following IP address**, and enter the **IP address** and **Subnet mask** for the Layer 4 cluster, as configured on Equalizer. Click **OK.**

   d. Click **OK** to return to **Network Connections.**

4. To configure the IIS HTTP server for DSR:

Open **Start > Administrative Tools > Internet Information Service (IIS) Manager**.

   a. In the left frame, expand the **local computer** and then **Web Sites** to display a list of the web sites running on the server.
   b. Right-click on the web site you want to configure for DSR and select **Properties.**
   c. On the **Web Site** tab, next to **IP address**, select the **Advanced** button.
   d. Select the **Add...** button under the top list box.
   e. Enter the **IP address** and the **TCP port** for the Layer 4 cluster, as configured on Equalizer. Click **OK**.
   f. Click **OK** twice to return to the **Internet Information Service (IIS) Manager.**

You should now be able to send client requests to the cluster IP and port, and get responses directly from the IIS HTTP server running on Windows 2003. Remember that static routes on your servers may be necessary, depending on your network configuration.

Adjusting ARP Behavior on Linux Servers

Some operating systems, such as Linux, will reply to ARP requests for the cluster IP address configured on the loopback interface. On such systems, the ARP behavior needs to be adjusted so the system only replies to ARP requests for IP addresses on non-loopback interfaces. The method used to do this varies between operating systems. For example, to do this on a Linux box, you would adjust specific kernel parameter values as shown below, by editing the file `/etc/sysctl`:

```
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.default.arp_ignore = 1
net.ipv4.conf.default.arp_announce = 2
net.ipv4.conf.eth0.arp_ignore = 1
net.ipv4.conf.eth0.arp_announce = 2
```

Configuring a Linux System running Apache for DSR

This is an example of how to configure a typical Linux system running Apache 2.0 for DSR:

1. Log into the Linux server as *root*, and enter the following command to configure a loopback interface:

```
    # ifconfig lo:dsr inet cluster-ip netmask 255.255.255.255
```

   Substitute the IP address of the DSR-enabled cluster on Equalizer for `cluster-ip` in the command above. Note that in most Linux distributions, you are configuring an alias for the loopback interface and should specify a netmask of 255.255.255.255 *instead of* the netmask used to configure the cluster on Equalizer.

2. Enter the following command to verify that the loopback alias was created:

```
# ifconfig lo:dsr
```

   The output should look like this:

```
lo:dsr Link encap:Local Loopback
inet addr:cluster-ip Mask:255.255.255.255
UP LOOPBACK RUNNING MTU:16436 Metric:1
```

3. To configure an Apache 2.0 server for DSR, edit the server configuration file to add a Listen directive for the cluster IP (on many systems, the configuration file is found at **/usr/local/etc/apache/httpd.conf**). Look for the first line beginning with the Listen directive, and add another line that looks like this:

```
Listen cluster-ip
```

   Where cluster-ip is the DSR-enabled cluster IP. Save your changes to the file.

4. Reboot the Apache server:

```
# apachectl restart
```

   You should now be able to send client requests to the cluster IP and port, and get responses directly from the Apache server running on Linux. Remember that static routes on your servers may be necessary, depending on your network configuration.

Configuring a Loopback Interface on Other Systems for DSR

The commands and interfaces used to configure a loopback interface vary slightly between operating systems, and sometimes between versions of the same operating system. Check the documentation for your server operating system for instructions on how to configure a loopback interface. For example, on some BSD systems, the command used in Step 1 in the previous section would be slightly different, as shown below:

```
# ifconfig lo0 cluster-ip netmask cluster-netmask alias
```

Notice that in this case, the netmask used matches the netmask used to configure the cluster on Equalizer, instead of 255.255.255.255 as in the Linux system example.

Weak and Strong Host Models and DSR

Network interfaces on non-routing systems use either the “weak host” or “strong host” models for packet transmission and reception (these models are defined in RFC1122). In the “strong host” model, a system that is not acting as a router cannot send or receive any packets on a given interface unless the destination/source IP in the packet is assigned to the interface. In the “weak host” model, this restriction does not apply.

In order for DSR to work, the “weak host” model must be enabled on the server’s loopback interface, as well as the interface on which requests are received from Equalizer.

Most Linux and Unix systems default to the “weak host” model on all network interfaces, so no additional configuration is usually necessary. For example, on FreeBSD and NetBSD, this behavior is controlled by the setting of `sysctl net.inet.ip.check_interface`, which by default is set to 0 (“weak host”).

Windows XP and Windows 2003 use the “weak host” model on all IPv4 interfaces and the “strong host” model on all IPv6 interfaces, and this is not configurable.

Windows Vista and Windows 2008 support “strong host” by default on all interfaces, but this is configurable for individual interfaces. Use the following command to list interface status:

```
netsh interface [ ipv4 | ipv6 ] show interface
```

The following three command are an example of changing the mode to “weak host” for the LAN and loopback interfaces:

```
netsh interface ipv4 set interface "Local Area Connection" weakhostreceive=enabled
netsh interface ipv4 set interface "Loopback" weakhostreceive=enabled
netsh interface ipv4 set interface "Loopback" weakhostsend=enabled
```

The interface names used in quotes above must match the interface names that appear in the Windows **Network Connections** folder.

Chapter 23
Smart Control

Sections within this chapter include:

Smart Control Overview

The Smart Control feature allows you to define a common administrative function or, **Smart Event** that executes the function based on pre-set threshold values for system parameters and statistics. It is a method for administrators to configure the system to automatically perform functions that may be dependent on threshold values or timing.

Smart Control IS NOT something that makes Layer 4 or Layer 7 traffic management decisions or individual requests. It was designed to provide automated control framework for ADC administrators for the automatic execution of scheduled tasks such as a system back up, or tasks performed at specific time intervals, and can be used to execute functions based on operating environment or conditions.

Why PHP?

An automated function in Smart Control is called a Smart Event. An example of a Smart Event would be if you specify that when the number of active servers in a particular cluster falls below a certain value a currently quiesced server could become active. Smart Events are configured using PHP scripting. PHP (Hypertext Preprocessor) is a server-side scripting language designed for web development but can also be used as a general-purpose programming language as it is in this case.With Smart Control you can create custom PHP scripts that automatically execute the desired function. Custom extensions for PHP allow you read and write to Load Balancing objects.

PHP.net is a valuable resource for documentation, events, training, user groups, and recommended reading for all levels of PHP programming.

You can create Smart Events using PHP scripting with either the CLI or GUI.

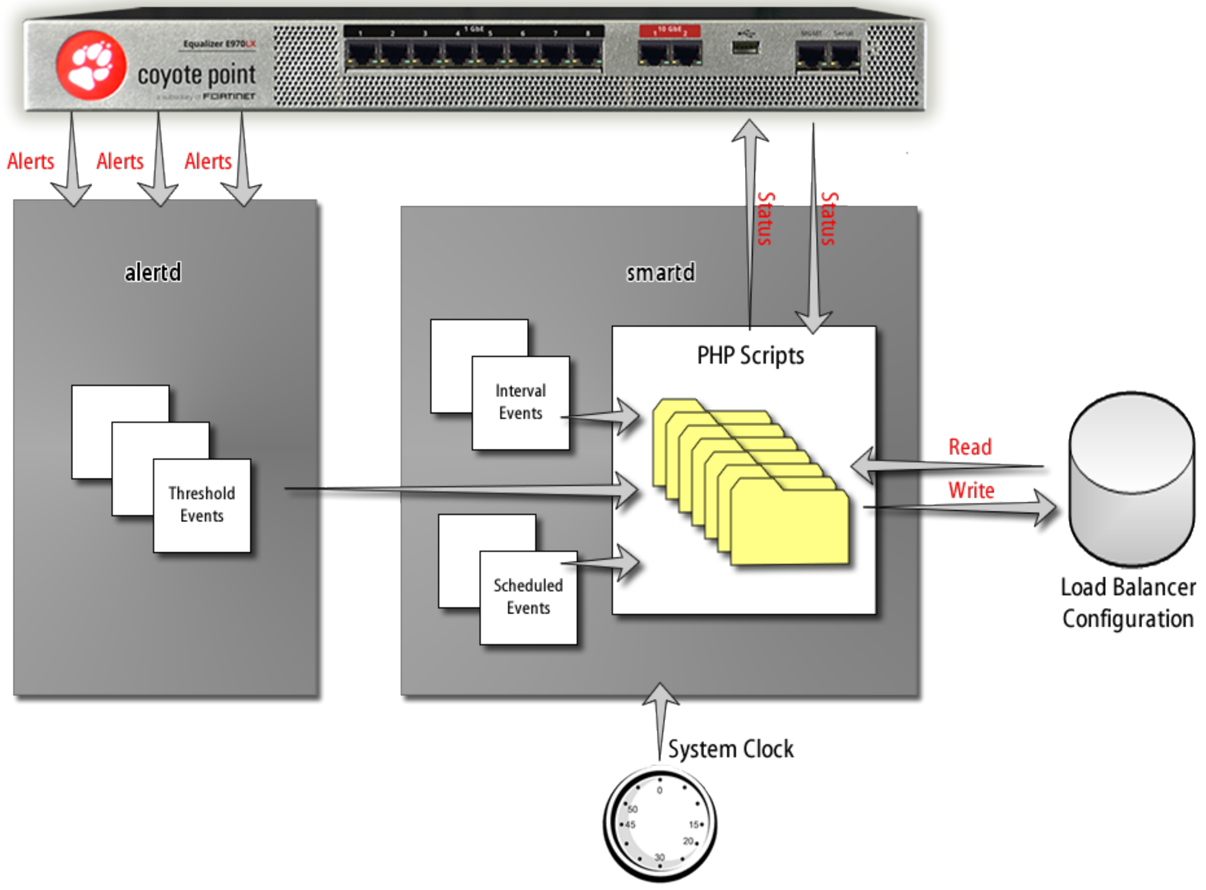
As a prerequisite for using Smart Control, it is highly recommended that you are familiar with PHP scripting and are knowledgeable of how load balancing objects are identified in the CLI.

How Smart Control Works

Automation framework drives the entire Smart Control infrastructure. This is managed by the Smart Control daemon-- smartd. This daemon loads configuration parameters from the configuration file and executes events as needed.

An alerts daemon—alertd, is responsible for Smart Control alerts. This daemon already cognizant of important events in the system, so it notifies smartd when a triggered type event has occurred so that smartd can execute the necessary script.

The figure below illustrates an overview of how Smart Control framework functions. The alertd daemon gets triggered event information from the system and then notifies the smartd daemon to execute a particular PHP script. The smartd daemon is responsible for running scripts when a particular scheduled or interval event occurs--based on a system clock. PHP scripts themselves can read and write system status and load balancer configuration information.



Smart Control Types

Your user account must have administrative privileges to create Smart Controls. They can execute scripts (events) in four ways:

1. They can be executed in real time using eqcli command line.syntax in the format `eqcli > smart_control` *`scname`* `run`.
2. They can be scheduled for a particular time using a cron-like mechanism. This is scheduled type.
3. They can be scheduled to repeat every several seconds. This is the **interval** type.
4. They can be set up to execute when an event occurs. This is **triggered** type. This kind of control is only run when a trigger event happens within the system. The alerting mechanism automatically notifies the alert daemon (alertd) when something has occurred. That daemon determines which alerts should be fired. For triggered events, the alertd daemon is responsible for the events themselves, rather than smartd. The alertd daemon reads the configuration and waits for triggers to occur. Once one does, it notifies smartd to execute a particular PHP script. Alerts need to be configured prior to configuring a Smart Control. Refer to "Configuring Alerts" on page 816 for descriptions.

Smart Control Configuration Guidelines

The Smart Control feature uses PHP as its underlying language. It is possible, however, to use this feature with minimal PHP knowledge. Some facts about Smart Control scripts:

- Each Smart Control script is wrapped inside of additional code prior to execution. The purpose of this is to enable added protection and save the environment when a script runs. As a result, tags such as “ `<?php ?>`” not needed.
- Smart Control scripts are intended to be run by the ADC in the background. That is, they are generally not designed to produce output. However, if a script produces output, either for debugging purposes, or because a PHP error has occurred, the output of the last time the script was run can be seen using the `lastrun` command. (See "Server Instance Class ("si") " on page 792 for a description).
- If a script produces a PHP execution error while running, an error is logged in the ADC log, however, the script continues to be executed as normal. The reason for this is that a different execution path may not produce the same error.
- By default, any variables declared during execution of a script are saved for the subsequent execution. If you would like to discard the environment between script executions, “`$save_ environment = FALSE;`” should be entered at the beginning of your script.
- Because Smart Control is based on PHP, most PHP constructs will work. These are generally for advanced users, however, they may be useful to create more robust scripts. For example, try/catch blocks can be used for any function that prompts an exception, as described below. However, although a script that catches these exceptions would be more robust, users who are not familiar with PHP may want to look at the output produced by the script, instead. (Exceptions will be displayed by the `lastrun` command).

Smart Control Classes

Each object in the ADC configuration is represented as a PHP class. The classes currently supported are:

- adc
- server
- srvpool
- si

Note - Support for additional PHP classes will be available in future releases.

A class variable is a variable defined in a class of which a single copy exists, regardless of how many instances of the class exist. There are two mechanisms to create a class variable:

1. To create a blank class variable, use the new keyword. For example: `$sp = new srvpool;`
2. To create a class variable that is filled in with the values of an existing ADC object, use the `getByName()` method. For example: `$sp = srvpool::getByName("sp00");`

Once a class variable exists, there are several methods to read information about that variable and common ways to modify the underlying object in the system configuration. Descriptions for each class are shown below.

The supported parameters for each class are the same as provided in the CLI. Flags in Smart Control are specified as individual Boolean parameters, rather than as a list (as in the CLI). For example, `$sp->custom_hc` is the custom health check value for a server pool, and `$sp->probe_ssl is a server pool's probe_ssl` flag.

Create PHP scripts for Smart Events using the File Editor. Refer to "Editing Files" on page 928for instructions on using it.

Server Pool Class (srvpool)

Parameters

The following are Server Pool parameters. Refer to Server Pool and Server Instance Commands for descriptions.

name (string)
acvq (string)
probe_maxtries (int)
probe_dto (int)
custom_actconn (int)
policy (string)
acvr (string)
probe_gto (int)
custom_hc (int)
probe_ssl (bool)
respv (int)
probe_interval (int)
probe_cto (int)
custom_delay (int)
disable (bool)

Methods

getByName(string name)

Description:
Fetch the server pool named 'name' from the configuration.

Returns:
On success: server pool object populated with all of its properties. On failure: An exception with a message and an error code.

Example:

```
    // Fetch a server pool named 'sp00'
    $sp = srvpool::getByName("sp00");
```

getInstanceByName(string name)

Description:
Fetch the server instance named 'name' which is part of this server pool from the configuration.

Returns:
On success: server instance object populated with all of its properties. On failure: An exception with a message and an error code.

Example:

```
    // Fetch a server pool named 'sp00' and then fetch its server instance 'si00'
    $sp = srvpool::getByName("sp00");
    $si = $sp->getInstanceByName("si00");
```

getInstanceList()

Description:
List server instances for this server pool from the configuration.

Returns: A map with the following keys:

`si_list`: list of server instance names as strings
`message`: a status message indicating success or failure of the operation
`status`: a status code: 0 indicates success, nonzero indicates failure

Example:

```
// Get list of server instances from server pool ‘sp00’
$sp = srvpool::getByName(“sp00”);
$list = $sp->getInstanceList();
// loop through the names in the “si_list”
for ($counter = 0; $counter < count($list["si_list"]); $counter++) {
// each name in the list is $list["si_list"][$counter]
}
```

getStatusDescription()

Description:
Get the status of this server pool as a string.

Returns:
A string containing the status of this server pool.

Example:

```
$sp = srvpool::getByName(“sp00”);
// print the status - accessible using ‘lastrun’ command.
echo $sp->getStatusDescription();
```

getStatusResp()

Description:
Get the status of this server pool as a numeric value.

Returns: A numeric value indicating the status:

0: There are no problems with this server pool.
1: There is an ‘informational’ status available, but the server pool is functional.
2: There is a problem with this server pool.

Example:

```
// If there is a problem with this pool, print the status (accessible using
‘lastrun’ command).
$sp = srvpool::getByName(“sp00”);
if ($sp->getStatusResp() == 2) {
        echo $sp->getStatusDescription();
}
```

stats(string statName)

Description:
Get the value of the statistic named '`statName`'. The available statistics are the same as those displayed in the CLI when using the `srvpool <name> stats` command.

Returns:
On success, the last-measured value of this statistic. On failure, an exception describing what went wrong: invalid statistic name or no statistic specified.

Example:
```
    $sp = srvpool::getByName("sp00");
    if ($sp->stats("TOTALPRCSD") > 100) {
           echo "Processed more than 100 requests through sp00";
    }
```

delete(optional Boolean forceFlag)

Description:
Delete this server pool. Can only be used on a server pool object which has been retrieved using `srvpool::getByName()`, and it must not have been modified since the last time that it was retrieved. If the server pool is in use by another object in the system, the flag `forceFlag` must be set to '`TRUE`' in order for it to be deleted.

Returns:
Map containing message string with a status code. Status code will be 0 if the deletion was successful and non-zero otherwise.

Example:
```
    $sp = srvpool::getByName("sp00");
    // Try to delete it, if it fails, force the deletion.
    $value = $sp->delete();
    if ($value["status"] != 0) {
       // Print out the reason for the failure, accessible using 'lastrun' command
       echo "Failed to delete because: ", $value["message"];
       $value = $sp->delete(TRUE);
    }
```

commit()

Description:
Push the changes to this server pool into the permanent configuration. If this server pool object was created using `getByName()`, this operation is treated as a modify. If it was created using the new keyword, it is treated as an addition.

Returns:
Map containing message string with a `status` code. Status code will be 0 if the commit was successful and non-zero otherwise.

Example:
```
    // Create a new server pool, then modify it
```

```
$sp = new srvpool;
$sp->name = "newsp";
$sp->commit();

// If we don’t do getByName(), the commit() below would fail with ‘object already
exists’ error because the system will try to add this object instead of modify it.
$sp = srvpool::getByName("newsp");
$sp->probe_maxtries = 2;
$sp->commit();
```

Server Class (server)

The following are Server parameters. Refer to Server Commands for descriptions.

Parameters

name (string)
proto (string)
probe_l3 (bool)
ip (string)
max_reuse_conn (int)
port (int)
reuse_conn_timeout (int)
bmc_addr (string)
bmc_user (string)
bmc_passwd (string)
bmc_cmd (string)

Methods

getByName(string name)

Description:
Fetch the server named `''name' from the configuration.

Returns:
On success: server object populated with all of its properties. On failure: An exception with a message and an error code.

Example:
```
    // Fetch a server named 'sv00'
    $sv = server::getByName("sv00");
```

getStatusDescription()

Description:
Get the status of this server as a string.

Returns:
A string containing the status of this server.

Example:
```
    $sv = server::getByName("sv00");
    // print the status - accessible using 'lastrun' command.
    echo $sv->getStatusDescription();
```

getStatusResp()

Description:
Get the status of this server as a numeric value.

Returns: A numeric value indicating the status:

0: There are no problems with this server.
1: There is an ‘informational’ status available, but the server is functional.
2: There is a problem with this server.

Example:
```
// If there is a problem with this server, print the status (accessible using
‘lastrun’ command).
$sv = server::getByName(“sv00”);
if ($sv->getStatusResp() == 2) {
        echo $sv->getStatusDescription();
}
```

getVid ()

Description:
Get the VID that the system thinks this server belongs on.

Returns:
A string value indicating the VID, or “unassigned” if no appropriate network is found.

Example:
```
// Print out ‘unknown’ or ‘known’ network, accessible using the ‘lastrun’ command.
$sv = server::getByName(“sv00”);
if ($sv->getVid() == "unassigned") {
echo "Unknown network";
} else {
       echo "Known network";
}
```

stats(string statName)

Description:
Get the value of the statistic named `‘statName’`. The available statistics are the same as those displayed in the CLI when using the server `<name>stats` command.

Returns:
On success, the last-measured value of this statistic. On failure, an exception describing what went wrong: invalid statistic name or no statistic specified.

Example:
```
$sv = server::getByName(“sv00”);
if ($sv->stats(“TOTALPRCSD”) > 100) {
        echo “Processed more than 100 requests through sv00”;
}
```

delete(optional Boolean forceFlag)

Description:
Delete this server. Can only be used on a server object which has been retrieved using server:: `getByName ()`, and it must not have been modified since the last time that it was retrieved. If the server is in use by another object in the system, the flag `forceFlag` must be set to `'TRUE'` in order for it to be deleted.

Returns:
Map containing message string with a status code. Status code will be 0 if the deletion was successful and non-zero otherwise.

Example:
```
    $sv = server::getByName("sv00");
    // Try to delete it, if it fails, force the deletion.
    $value = $sv->delete();
    if ($value["status"] != 0) {
        // Print out the reason for the failure, accessible using 'lastrun' command
        echo "Failed to delete because: ", $value["message"];
        $value = $sv->delete(TRUE);
    }
```

commit()

Description:
Push the changes to this server into the permanent configuration. If this server object was created using `getByName()`, this operation is treated as a modify. If it was created using the new keyword, it is treated as an addition.

Returns:
Map containing message string with a status code. Status code will be 0 if the commit was successful and non-zero otherwise.

Example:
```
    // Create a new server, then modify it
    $sv = new server;
    $sv->name = "newsv";
    $sv->proto = "tcp";
    $sv->ip = "1.2.3.4";
    $sv->port = 80;
    $sv->commit();

    // If we don't do getByName(), the commit() below would fail with 'object already
    exists' error because the system will try to add this object instead of modify it.
    $sv = server::getByName("newsv");
    $sv->port = 443;
    $sv->commit();
```

Server Instance Class (si)

The following are Server Instance parameters. Refer to Server Pool and Server Instance Commands for descriptions.

Parameters

name(string)
probe_port (int)
quiesce (bool)
sp (object): The server pool that this server instance belongs to, as retrieved using `srvpool::getByName()`.
weight (int)
hot_spare (bool)
probe_l4 (bool)
maxconn (int)
persist_override (bool)
strict_maxconn (bool)

Methods

getByName(object srvpool, string name)

Description:
Fetch the server instance named `'name'` from the server pool object `srvpool`. The server pool object should have been previously fetched with `srvpool::getByName()`.

Returns:
On success: server instance object populated with all of its properties. On failure: An exception with a message and an error code.

Example:
```
    // Fetch a server instance named ‘sv00’
    $sp = srvpool::getByName(“sp00”);
    $si = si::getByName($sp, "sv00");
```

getStatusDescription()

Description:
Get the status of this server instance as a string.

Returns:
A string containing the status of this server instance.

Example:
```
    $sp = srvpool::getByName(“sp00”);
    $si = si::getByName($sp, "sv00");
    // print the status - accessible using ‘lastrun’ command.
    echo $si->getStatusDescription();
```

getStatusResp()

Description:
Get the status of this server instance as a numeric value.

Returns: A numeric value indicating the status:

0: There are no problems with this server instance.
1: There is an 'informational' status available, but the server instance is functional.
2: There is a problem with this server instance.

Example:
```
// If there is a problem with this server instance, print the status (accessible
using 'lastrun' command).
$sp = srvpool::getByName("sp00");
$si = si::getByName($sp, "sv00");
if ($si->getStatusResp() == 2) {
        echo $si->getStatusDescription();
}
```

getCurrentWeight()

Description:
Get the dynamic weight of this server instance as a numeric value.

Returns:
A numeric value indicating the weight.

Example:
```
// Print the current weight (accessible using 'lastrun' command).
$sp = srvpool::getByName("sp00");
$si = si::getByName($sp, "sv00");
if ($si->getStatusResp() == 2) {
        echo $si->getCurrentWeight();
}
```

stats(string statName)

Description:
Get the value of the statistic named `'statName'`. The available statistics are the same as those displayed in the CLI when using the srvpool <name> si <name> stats command.

Returns:
On success, the last-measured value of this statistic. On failure, an exception describing what went wrong: invalid statistic name or no statistic specified.

Example:
```
$sp = srvpool::getByName("sp00");
$si = si::getByName($sp, "sv00");
if ($si->stats("TOTALPRCSD") > 100) {
        echo "Processed more than 100 requests through sv00";
}
```

delete(optional Boolean forceFlag)

Description:
Delete this server. Can only be used on a server instance object which has been retrieved using `si::getByName()`,and it must not have been modified since the last time that it was retrieved. If the server instance is in use by another object in the system, the flag `forceFlag` must be set to '`TRUE`' in order for it to be deleted.

Returns:
Map containing message string with a status code. Status code will be 0 if the deletion was successful and non-zero otherwise.

Example:
```
    $sp = srvpool::getByName("sp00");
    $si = si::getByName($sp, "sv00");
    // Try to delete it, if it fails, force the deletion.
    $value = $si->delete();
    if ($value["status"] != 0) {
        // Print out the reason for the failure, accessible using 'lastrun' command
        echo "Failed to delete because: ", $value["message"];
        $value = $si->delete(TRUE);
    }
```

commit()

Description:
Push the changes to this server instance into the permanent configuration. If this server object was created using `getByName()`, this operation is treated as a modify. If it was created using the new keyword, it is treated as an addition.

Returns:
Map containing message string with a `status` code. Status code will be 0 if the commit was successful and non-zero otherwise.

Example:
```
    // Create a new server, then modify it
    $sp = srvpool::getByName("sp00");
    $si = new si;
    $si->sp = $sp;
    $si->name = "sv00";
    $si->weight = 10;
    $si->commit();
    // If we don't do getByName(), the commit() below would fail with 'object already
    exists' error because the system will try to add this object instead of modify it.
    // The server pool must be re-fetched also since it was modified when we added a
    server instance.
    $sp = srvpool::getByName("sp00");
    $si = $si = si::getByName($sp, "sv00");
    $si->weight = 100;
    $si->commit();
```

ADC Class (adc)

Parameters

An ADC class currently has no publicly accessible parameters.

Methods

cli(string command)

Description:
Run the provided *command* exactly as it would be if entered into the CLI.

Returns:
Map containing `cli_buf` string with a *status* code. Status code will be 0 if the CLI command was successful and non-zero otherwise. `cli_buf` string will contain the output that would have been displayed on the screen if this command were typed into the CLI. Additional message string contains description of the status code.

Example:

```
// if there are fewer than 10 active connections, rebalance the failover peers
$sp = srvpool::getByName(“sp00”);
if ($sp->stats(“ACTIVECONX”) < 10) {
       adc::cli(“rebalance”);
}
```

getSrvpoolList()

Description:
List server pools from the configuration.

Returns:
A map with the following keys:

`srvpool_list`: list of server pool names as strings
`message`: a status message indicating success or failure of the operation
`status`: a status code: 0 indicates success, nonzero indicates failure

Example:

```
// Get list of server pools
$list = adc::getSrvpoolList();
// loop through the names in the “srvpool_list”
for ($counter = 0; $counter < count($list["srvpool_list"]); $counter++) {
       // each name in the list is $list["srvpool_list"][$counter]
}
```

getServerList()

Description:
List servers from the configuration.

Returns:

A map with the following keys:

`server_list`: list of server names as strings
`message`: a status message indicating success or failure of the operation
`status`: a status code: 0 indicates success, nonzero indicates failure

Example:
```
// Get list of servers
$list = adc::getServerList();
// loop through the names in the "server_list"
for ($counter = 0; $counter < count($list["server_list"]); $counter++) {
       // each name in the list is $list["server_list"][$counter]
}
```

ipmi

Description:
Issue a power on/power off/shutdown/reset command to a BMC server in the following sequence - IP Address, User name, Password, Command.

Returns:
Output buffer with message string and status (error) code. The status code will be 0 if the ipmi command was successful otherwise it will be non-zero. In case of error, the out_buf string will contain the output returned from ipmitool else it will be blank. Additional message string contains a description of the status code.

Example:
```
//There are two servers attached to the server pool. If there are more than 1000
connections on the server pool, then switch on server-2 and if the connections
drop to 500, then switch off server-2. This is can be implemented on peak traffic,
where the maximum number of servers should be UP and on non-peak traffic, where
the minimum number of servers should be UP.

$sp = srvpool::getByName("sp00");

if ($sp->stats("ACTIVECONX") > 1000) {
$ret = adc::ipmi("1.1.1.1","root","abcd","poweron");
} else if ($sp->stats("ACTIVECONX") < 500) {
$ret = adc::ipmi("1.1.1.2","root","xyz","poweroff");
}
```

Sample Trigger Script for the Configuration of Multiple Hot Spare Servers

The following is an example of a trigger script that allows the configuration of multiple hot spare servers. The purpose of this is to monitor the active servers; and if they both go down or become unavailable, the hot spare servers will become active:

```
$commitChanges = FALSE;
$sp = srvpool::getByName(“sp00”);
$sv00 = si::getByName($sp, “sv00”);
$sv01 = si::getByName($sp, “sv01”);
$sv02 = si::getByName($sp, “sv02”);
$sv03 = si::getByName($sp, “sv03”);
// if both of these servers are down, make the other two servers active and
hot_spare these.
if ($sv00->getStatusResp() == 2 && $sv01->getStatusResp() == 2) {
       $sv00->hot_spare = TRUE;
       $sv01->hot_spare = TRUE;
       $sv02->hot_spare = FALSE;
       $sv03->hot_spare = FALSE;
       $commitChanges = TRUE;
} else if ($sv02->getStatusResp() == 2 && $sv03->getStatusResp() == 2) {
       $sv00->hot_spare = FALSE;
       $sv01->hot_spare = FALSE;
       $sv02->hot_spare = TRUE;
       $sv03->hot_spare = TRUE;
       $commitChanges = TRUE;
}
// avoid unnecessary commits by using a flag
if ($commitChanges == TRUE) {
        $sv00->commit();
        $sv01->commit();
        $sv02->commit();
        $sv03->commit();
}
```

Sample Trigger Script for Rebooting the System

The following is an example of a trigger script that will reboot the system (causing a failover) if a critical IP address cannot be reached or"‶pinged". It should be noted that the script uses string parsing and will consume a fair amount of CPU resources. It is recommended that you use this type of script if no other mechanisms are available.

```
$out = adc::cli("ping 10.0.0.68");
// if the ping fails, the string “0 packets received” is output, so look
for that
if (strstr($out['cli_buf'], "0 packets received")) {
       adc::cli(“reboot”);
}
```

Adding Smart Controls

Smart Controls can be added using the CLI or GUI. If you associated an alert with a Smart Control, the means which you selected to notified of the alert will be displayed.

Adding a Smart Control using the CLI

1. Enter the following to assign a name to the smart event.

```
eqcli > smart_controlsmart_control_name
```

2. Activate the editor by entering the following in the smart control context.

```
eqcli > script edit/URL
```

   where:

   edit invokes the script editor to enter/create the desired script

   URL fetches the script from the entered fully qualified ftp/http site

   If you upload a script and it fails syntax checking, the smart control is automatically disabled and a message will be displayed:

   "*Smart Control is disabled because it failed syntax checking. Re-uploading the script, or editing it to fix the syntax error would resolve the problem.*"

3. Create/edit the script in the script editor and save the script to the data store.
4. Determine whether the script is to be run by schedule or at an interval by entering either of the following.
   a. Running the script at an interval will run the selected script at the **interval** assigned.

```
eqcli sc-scname > interval seconds
```

   where ***seconds*** is the number of seconds at which the script will run.

b. Setting up a **`schedule`** will run the script at the times that you schedule .The **`schedule`** for Smart Control is in the local time zone. Therefore, if you set it to run at 12:00, this means 12:00 of whatever timezone the system is set to and not GMT or UTC.

```
eqcli sc-scname > schedule schedule string
```

where ***`schedule string`*** is in the standard cron format, but with an additional first column:

```
second:        0-59
minute:        0-59
hour:          0-23
day of month:  1-31
month:         1-12 (or names, see below)
day of week:   0-7 (0 or 7 is Sun, or use names below)
"Lists" are supported (using comma) but "steps" (generally
specified with a "/") are not supported.  White space (' ' or
\t) is a  column break. The parsing starts at the first non-
white-space character. If the wrong number of columns (not 6)
is parsed, parsing fails.
Day names:   Mon,Tue,Wed,Thu,Fri,Sat,Sun
Month names: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep,
Oct, Nov, Dec
```

Note - The schedule string must be enclosed in quotes. i.e.: "* 0,30 * * * Mon" would be translated as 'Every Monday, run this this every 30 seconds'.

Displaying Configured Smart Controls

To display existing Smart Controls enter `eqcli >` **`show smart_control`*`schedule_sc_name`***. The following is an example of a scheduled smart control.

```
eqcli > show smart_control schedule_sc_name

Last Execution Time: Fri Feb 20 16:37:45 UTC 2015
Next Execution Time: N/A
Smart Control Name  : test
Schedule            :* * 1 2 Feb Mon
Interval            : 0
Run Limit           : 10
Flags               :
Script              :
Name    Interval  Schedule
Test_1  648000
Test_2  0         * * 1 2 Feb Mon
eqcli >
```

Running a Script Manually Using the CLI

After creating a script you can run the script manually through the CLI. Enter the following:

```
eqcli > smart_control run scname
```

To see when the Smart Control was last executed, use the following format:

```
eqcli > show smart_control smartcontrlname
```

where ***smartcontrolname*** is the name of the smart control.

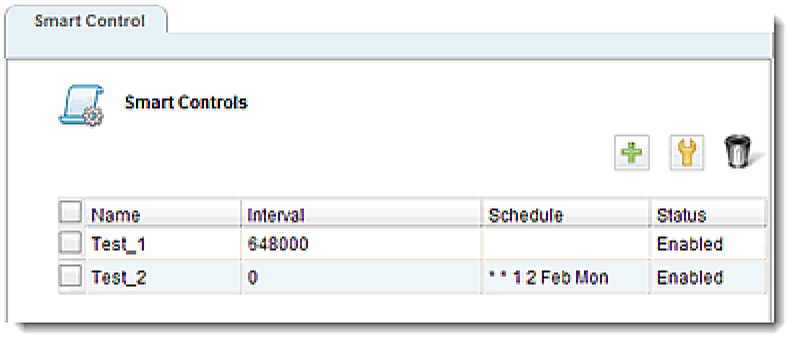
The following is an example:

```
eqcli > show smart_control sc-test
This Smart Control is enabled.
Last Execution Time: 11/17/2014 17:07:55
Next Execution Time: 11/18/2014 17:07:55
Smart Control Name  : sc-test
Schedule            :
Interval            : 648000
Run Limit           : 10
Flags               :
Script              :
echo "test smart control";
```

Refer to "Smart Control" on page 217 for a complete list of CLI Smart Control commands.

Adding or Modifying a Smart Control using the GUI

1. Configure an alert as described in "Configuring Alerts" on page 816.You must set the **Alert Notification Type** on the **Add Alert** configuration screen to "**smartd**" for the Smart Control to be executed when the alert is made by the system.
2. Log in to the GUI..
3. Click on **System> Global > Smart Controls** on the left navigational pane.
4. Click on **Smart Controls** to display the configured Smart Controls Summary screen. An example is shown below.



Smart Control

Smart Controls

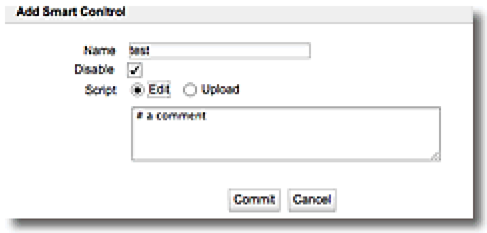
Name	Interval	Schedule	Status
Test_1	648000		Enabled
Test_2	0	* * 1 2 Feb Mon	Enabled

5. Do one of the following to add a new Smart Control:

   a. Click on + to add a new smart event. To modify an existing Smart Control either double click one from the list or select one and click on the . In both cases the **Modify Smart Control** dialogue screen will be displayed. The dialogue screen will be change, based on your choice of Smart Control types. In the example below, an **Interval** Smart Control Type is selected. If you are modifying a Smart Control, the **Name** of the Smart Control will be displayed at the top of the screen and the current settings will be displayed.

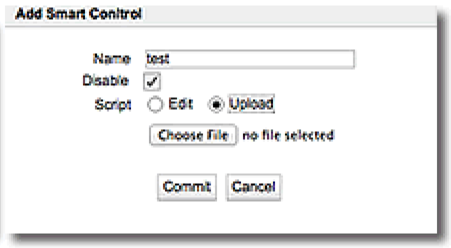
   b. Right click on **Smart Control** on the left navigational pane and select **Add Smart Control**.

Either method will display the **Add Smart Control** dialogue as shown below. Note that there are two options: The first option is to manually enter a script in the area provided. This option is available with the **Edit** option is selected. The second option is to upload a local script file (.txt). This option is available when the **Upload** option is selected.


Add Smart Control

Name: test
Disable: ✓
Script: (•) Edit ( ) Upload

```
# a comment
```


Add Smart Control

Name: test
Disable: ✓
Script: ( ) Edit (•) Upload

no file selected

6. Enter a name for the Smart Control in the **Name** box.
7. By default, all new Smart Controls are **Disabled** by default. If you want to Enable it immediately, click on the **Disable** check box to remove the √.This option is more commonly used when a Smart Control has been previously configured and you simply want to disable it.
8. Enter a script in the space provided if you have selected the **Edit** option. If using the Upload option, click on the **Choose File** button and follow the dialogue to upload a local script file to your Equalizer file store. Click on **Commit** to save the Smart Control. The Smart Control configuration screens will then be available.

If you upload a script and it fails syntax checking, the smart control is automatically disabled and a message will be displayed:

"*Smart Control is disabled because it failed syntax checking. Re-uploading the script, or editing it to fix the syntax error would resolve the problem.*"

Displaying the Smart Control Configuration Screens

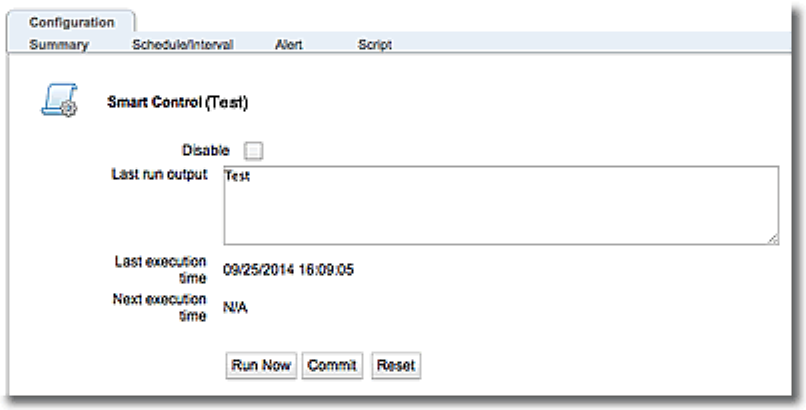
Smart Control configuration screens include the **Configuration Summary**, **Schedule/Interval**, **Alert**, and **Script**. They are all displayed when:

- a new Smart Control is entered
- a configured Smart Control is selected from the **Smart Control** branch on the left navigational pane.
- you double click a Smart Control or select a Smart Control and select .from the Smart Control Summary screen described in step 5.

Configuration Summary

The following is an example of the **Configuration > Summary** screen. From this screen, you have the option of:

- Enabling or Disabling the Smart Control using the check box. Click on the **Commit** button if you have changed this setting.
- Viewing the **Last run output**. This is.will display the output of the Smart Control daemon from the last run. The complete output of the **Last run output** display may not be displayed as the output of this is limited to 1023 characters.
- Viewing the **Last** (script) **execution time** (and date).
- Viewing the **Next execution time**. This is applicable when an interval or schedule has been configured. Refer to the descriptions that follow for details.
- Running the Smart Control by clicking on the **Run Now** button. This will run the script immediately.



Disable ☐
Last run output: Test
Last execution time: 09/25/2014 16:09:05
Next execution time: N/A

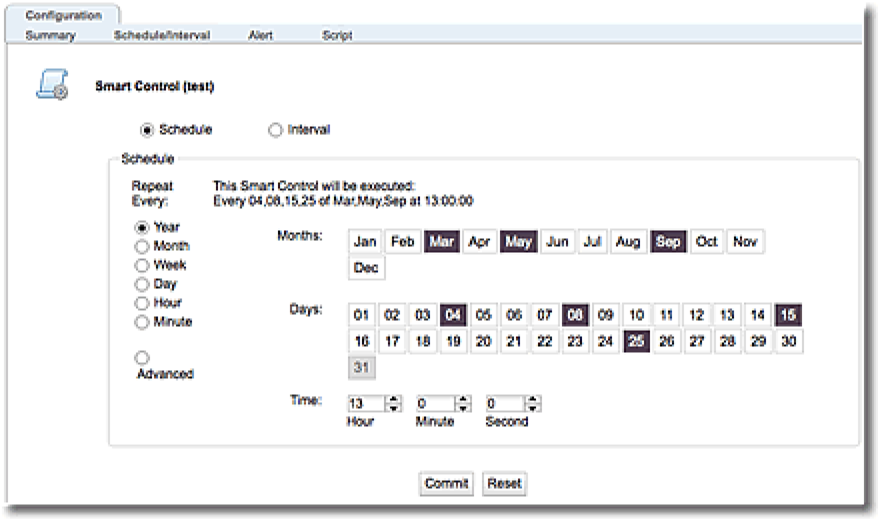
If you click on the **Reset** button, all of the information on the screen will be reset.

Schedule/Interval Configuration

The following is an example of the **Configuration > Schedule/Interval** screen. From this screen you have the option of configuring the Smart Control to be executed using the fine-grained scheduling tools and options or you can configure it to be executed at regular intervals. To using the Schedule feature, click on the **Schedule** option and then select the frequency option that you want to use in the **Repeat Every** column. To use the Interval option, click on the Interval option and configure the **Run Every** options that you require:

Year

The Yearly configuration screen will be displayed when you select **Schedule** and then select the **Year** option. These scheduling options allow you to configure a Smart Control to be executed on a monthly basis, by date and hour (24 hour clock), minutes and seconds. In the example below, the Smart Control is scheduled to be executed on the 4th, 8th, 15th and 25th of March, May, and September at 13:00. (1PM).


Smart Control (test)

[✓ Schedule] Interval

Schedule

Repeat Every: [✓ Year] Month Week Day Hour Minute Advanced

This Smart Control will be executed:
Every 04,08,15,25 of Mar,May,Sep at 13:00:00

Months: Jan Feb [✓ Mar] Apr [✓ May] Jun Jul Aug [✓ Sep] Oct Nov Dec

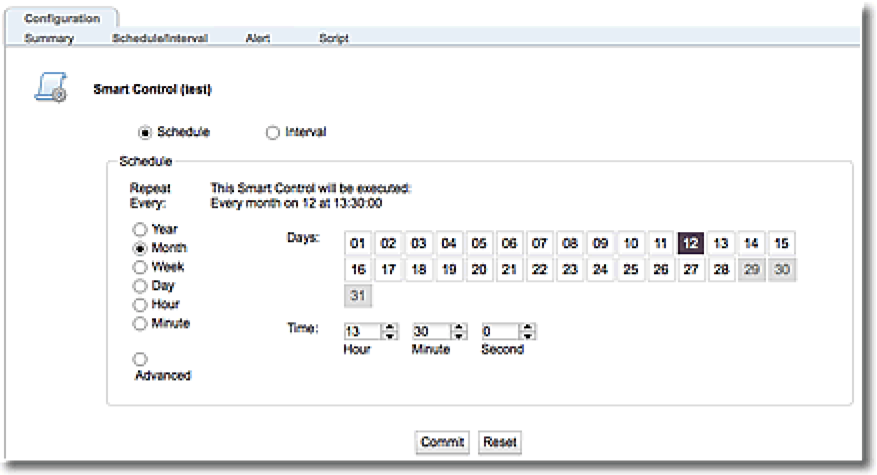
Days: 01 02 03 [✓ 04] 05 06 07 [✓ 08] 09 10 11 12 13 14 [✓ 15] 16 17 18 19 20 21 22 23 24 [✓ 25] 26 27 28 29 30 31

Time: Hour 13 Minute 0 Second 0

1. Select the **Months** and **Days** by clicking them. You can select one or all of the months or days. Click on a previously selected month or day to unselect it.
2. Select the Smart Control execution **Time** by using the **Hour**, **Minute** , and **Second** selectors.
3. Click on **Commit** to save the Scheduling options.

Month

The Monthly configuration screen will be displayed when you select **Schedule** and then select the **Month** option. This scheduling options allows you to configure a Smart Control to be executed on specific days of the month and the times by hour (24 hour clock), minutes and seconds. In the example below, the Smart Control is scheduled to be executed every month on the 12th of the month at 13:00 (1PM).



Smart Control (test)

Schedule [✓ Schedule] Interval

Schedule

Repeat Every: This Smart Control will be executed: Every month on 12 at 13:30:00

Year [✓ Month] Week Day Hour Minute Advanced

Days: 01 02 03 04 05 06 07 08 09 10 11 [✓ 12] 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Time: 13 Hour 30 Minute 0 Second

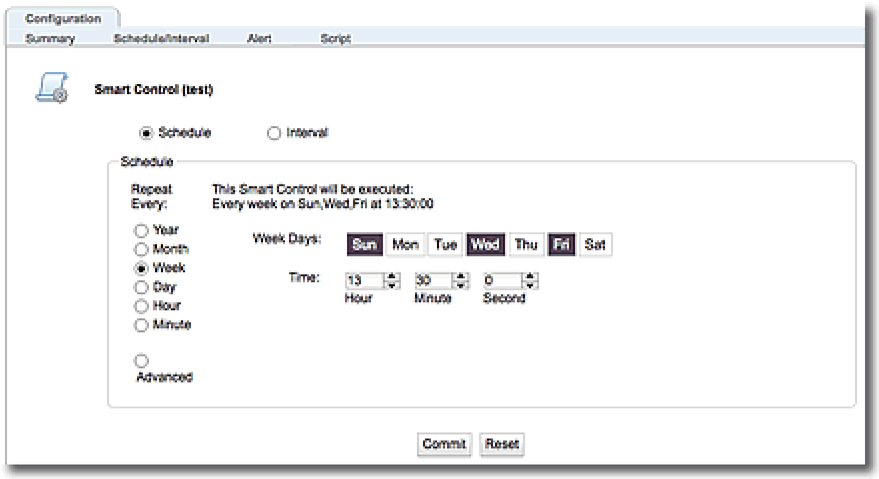
1. Select the **Days** by clicking them. You can select one or all of the days. Click on a previously selected day of the month to unselect it.

Note - You will note that there are 28 days on the monthly schedule. To configure Smart Control execution on the 30th or 31st of the month, use the Yearly scheduling option described above and configure Smart Control execution for the 30th and or 31st, as necessary.

2. Select the Smart Control execution **Time** by using the **Hour**, **Minute** , and **Second** selectors.
3. Click on **Commit** to save the Scheduling options.

Week

The Weekly configuration screen will be displayed when you select **Schedule** and then select the **Week** option. This scheduling options allows you to configure a Smart Control to be executed on specific days of the week and the times by hour (24 hour clock), minutes and seconds. In the example below, the Smart Control is scheduled to be executed every Sunday, Wednesday, and Friday at 13:00 (1PM).



This Smart Control will be executed:
Every week on Sun,Wed,Fri at 13:30:00

(●) Schedule ( ) Interval

Repeat Every: ( ) Year ( ) Month (●) Week ( ) Day ( ) Hour ( ) Minute ( ) Advanced

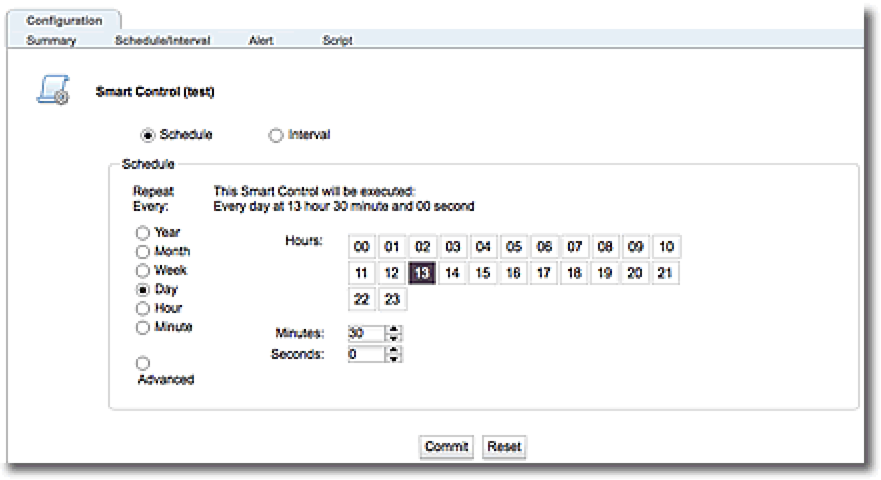
Week Days: [✓ Sun] Mon Tue [✓ Wed] Thu [✓ Fri] Sat

Time: 13 Hour, 30 Minute, 0 Second

1. Select the **Week Days** by clicking them. You can select one or all of the days of the week. Click on a previously selected Week Day to unselect it.
2. Select the Smart Control execution **Time** by using the **Hour**, **Minute** , and **Second** selectors.
3. Click on **Commit** to save the Scheduling options.

Day

The Daily configuration screen will be displayed when you select **Schedule** and then select the **Day**option. This scheduling options allows you to configure a Smart Control to be executed daily at times by hour (24 hour clock), minutes and seconds. In the example below, the Smart Control is scheduled to be executed every day at 13:00 (1PM).



Smart Control (test)

(● Schedule) ○ Interval

Schedule

Repeat Every: This Smart Control will be executed: Every day at 13 hour 30 minute and 00 second

○ Year ○ Month ○ Week [✓ Day] ○ Hour ○ Minute ○ Advanced

Hours: 00 01 02 03 04 05 06 07 08 09 10 11 12 [✓ 13] 14 15 16 17 18 19 20 21 22 23

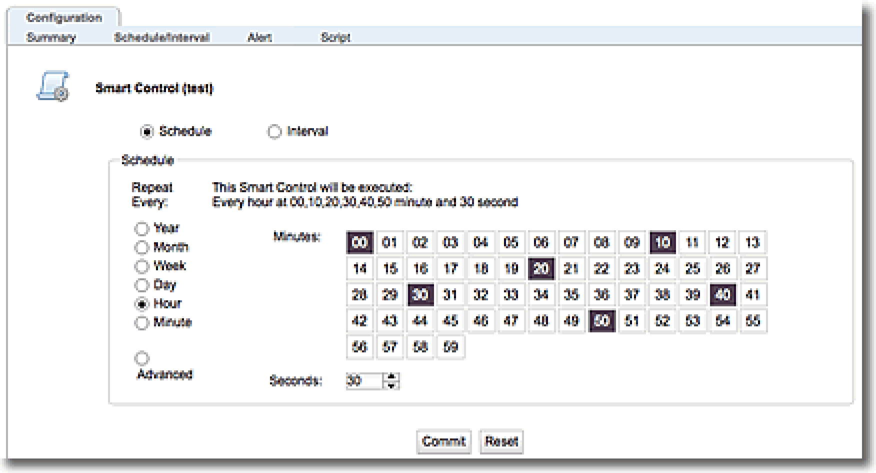
Minutes: 30

Seconds: 0

1. Select the **Hours**by clicking them. You can select one or all of the **Hours.** Click on a previously selected day of the **Hour** to unselect it.
2. Click on **Commit** to save the Scheduling options.
3. Use finer grained timing by using the **Minute** , and **Second** selectors in addition to selecting **Hours.**

Hour

The Hourly configuration screen will be displayed when you select **Schedule** and then select the **Hour**option. This scheduling options allows you to configure a Smart Control to be executed daily and hourly by minutes and seconds. In the example below, the Smart Control is scheduled to be executed every hour at 10 minutes and 30 seconds, 20 minutes and 30 seconds, 30 minutes and 30 seconds, 40 minutes and 30 seconds, and 50 minutes and 30 seconds.



Smart Control (test)

[✓ Schedule] Interval

Schedule

Repeat Every: Year Month Week Day [✓ Hour] Minute Advanced

This Smart Control will be executed:
Every hour at 00,10,20,30,40,50 minute and 30 second

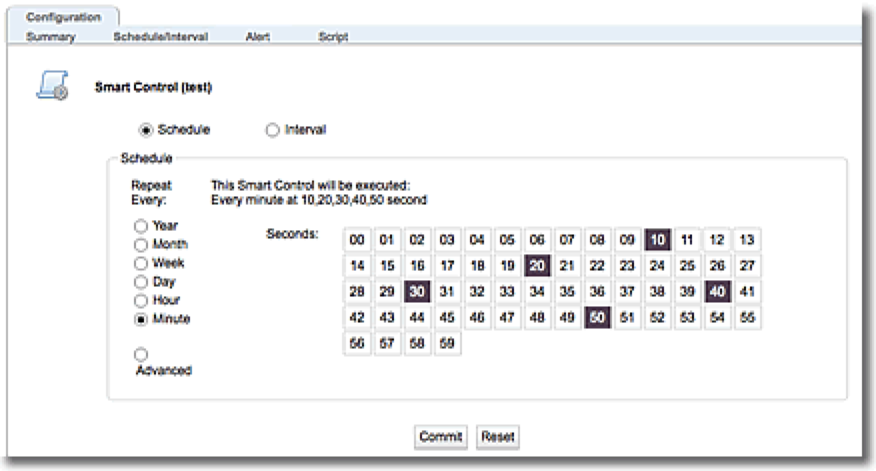
Minutes: [✓ 00] 01 02 03 04 05 06 07 08 09 [✓ 10] 11 12 13 14 15 16 17 18 19 [✓ 20] 21 22 23 24 25 26 27 28 29 [✓ 30] 31 32 33 34 35 36 37 38 39 [✓ 40] 41 42 43 44 45 46 47 48 49 [✓ 50] 51 52 53 54 55 56 57 58 59

Seconds: 30

1. Select the **Minutes** by clicking them. You can select one or all minutes . Click on a previously selected **Minute** to unselect it.
2. Use the **Seconds** selector if necessary.
3. Click on **Commit** to save the Scheduling options.

Minute

The Minute configuration screen will be displayed when you select **Schedule** and then select the **Minute**option. This scheduling options allows you to configure a Smart Control to be executed by seconds. In the example below, the Smart Control is scheduled to be executed every minute at hour, 10 seconds, 20 seconds, 30 seconds, 40 seconds, and 50 seconds.



Smart Control (test)

[✓ Schedule] Interval

Schedule

Repeat Every: Year Month Week Day Hour [✓ Minute] Advanced

This Smart Control will be executed:
Every minute at 10,20,30,40,50 second

Seconds: 00 01 02 03 04 05 06 07 08 09 [✓ 10] 11 12 13 14 15 16 17 18 19 [✓ 20] 21 22 23 24 25 26 27 28 29 [✓ 30] 31 32 33 34 35 36 37 38 39 [✓ 40] 41 42 43 44 45 46 47 48 49 [✓ 50] 51 52 53 54 55 56 57 58 59

1. Select the **Minutes**by clicking them. You can select one or all **Minutes.** Click on a previously selected day of the **MInutes**to unselect it.

Advanced

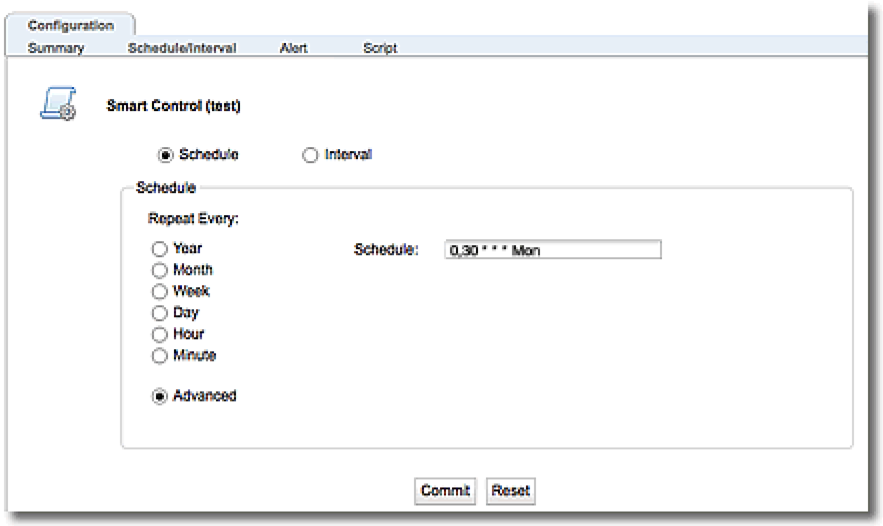
The Advanced configuration screen will be displayed when you select **Schedule** and then select the **Advanced**option. This scheduling options allows you to configure a Smart Control to be executed in a manually entered, string format in standard cron format, but with an additional first column -- second.

Lists and ranges are supported (use ',' as a list delimiter or '-' as a range delimiter), but steps are not. White space (' ' or) is a column break, and consecutive white spaces are treated as one. Fields which are an asterisk ('*') are skipped.

The schedule string must be enclosed in quotes.

seconds	0-59
minutes	0-59
hour	0-23
day of month	1-31
month	1-12 (or Day/Month names as shown below)
day of week	0-7 (0 or 7 is Sun, or use names)
Day names	'Mon','Tue','Wed','Thu','Fri','Sat','Sun' Month names: 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'

In the example below, the string **"* 0,30 * * * Mon"** means to execute the Smart Control *Every Monday, every 30 seconds.*



Smart Control (test)

[✓ Schedule] Interval

Schedule

Repeat Every:

Year
Month
Week
Day
Hour
Minute

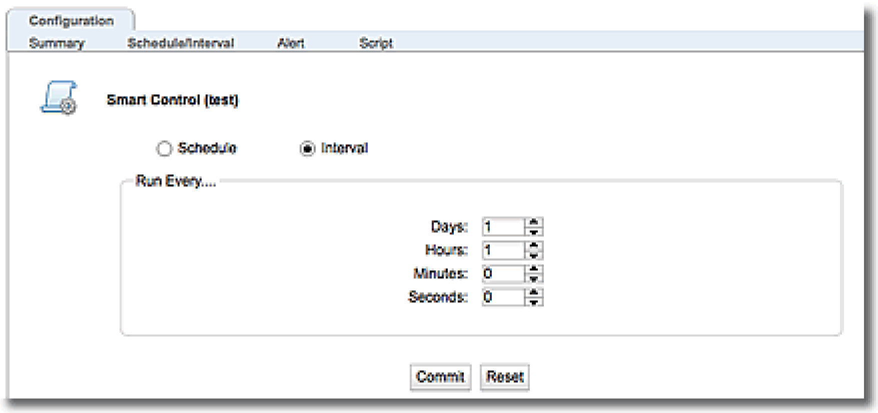
[✓ Advanced]

Schedule: 0,30 * * * Mon

1. Enter the string in the **Schedule** space in the format described above.
2. Click on **Commit** to save the Scheduling options.

Interval

The Interval configuration screen will be displayed when you select the **Interval**. This scheduling options allows you to configure a Smart Control to be executed at regular intervals. In the example below, the Smart Control is scheduled to be executed every day, every hour.



Smart Control (test)

Schedule [✓ Interval]

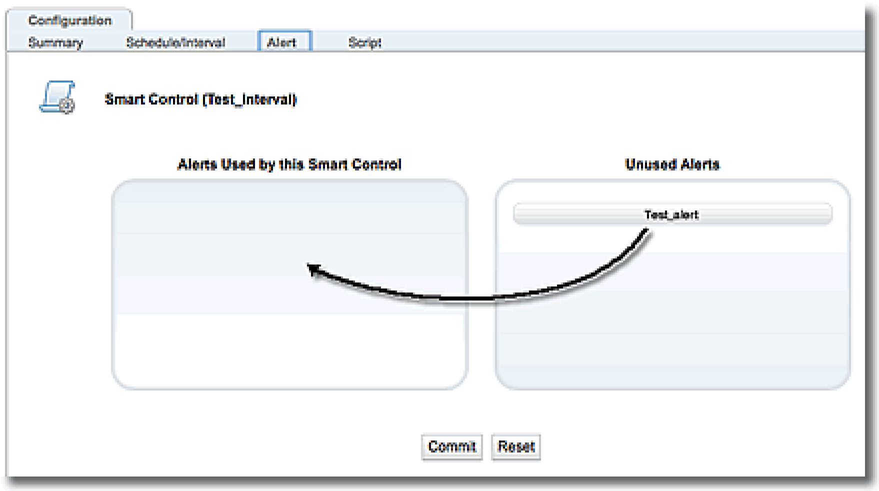
Run Every....

Days: 1
Hours: 1
Minutes: 0
Seconds: 0

1. Select the interval to use by using the Days, Hours, **Minutes** , and **Seconds** selectors.
2. Click on **Commit** to save the **Interval** options.

Attaching Alerts

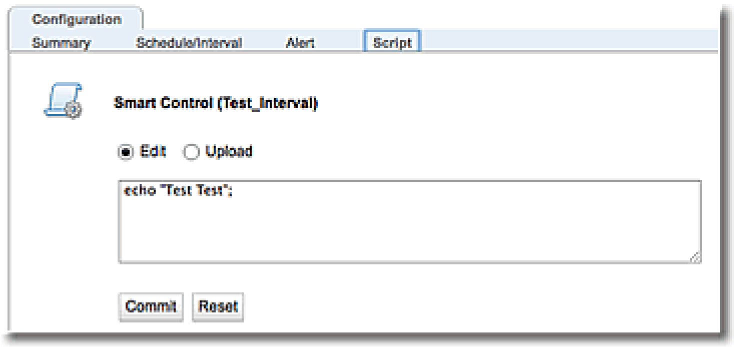
Selecting the **Alert** tab displays the **Configuration > Alerts** screen shown below. An alerts daemon is responsible for Smart Control alerts. This daemon is cognizant of important events in the system, so it generates an alert for which a Smart Control is configured. The **Configuration > Alert** configuration screen is where an alert is attached to the Smart Control.



1. Verify that you have configured an alert as described in "Configuring Alerts" on page 816.You must set the **Alert Notification Type** on the **Add Alert** configuration screen to "**smartd**" for the Smart Control to be executed when the alert is made by the system.
2. Drag and drop an alert from the **Unused Alerts** pane to the **Alerts Used by this Smart Control** pane.
3. Click on **Commit** to save the Alert.

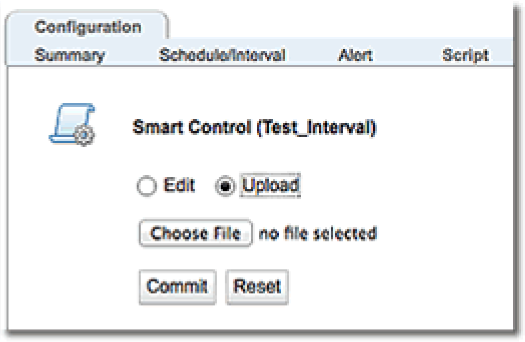
Script Edit

Selecting the **Script** tab displays the editing screen, which is virtually identical to the **Add Smart Control** dialogue screens. It displays either the manually entered script or the contents of the uploaded .txt file. You can edit scripts in this configuration screen. Since the Smart Control is already generated, there is no need to enter a name.



[✓ Edit] Upload

```
echo "Test Test";
```



Edit [✓ Upload]

1. **Edit** the script as necessary, Disable it, or **Upload** a new script file. If using the **Upload** option, click on the **Choose File** button and follow the dialogue to upload a local script file to your Equalizer file store.
2. Click on **Commit** to save the changes to the Smart Control.

Displaying the Output of the Last Smart Control Run

The first 1023 characters of the output generated during the previous run of a smart control can be displayed in both the CLI and the GUI.

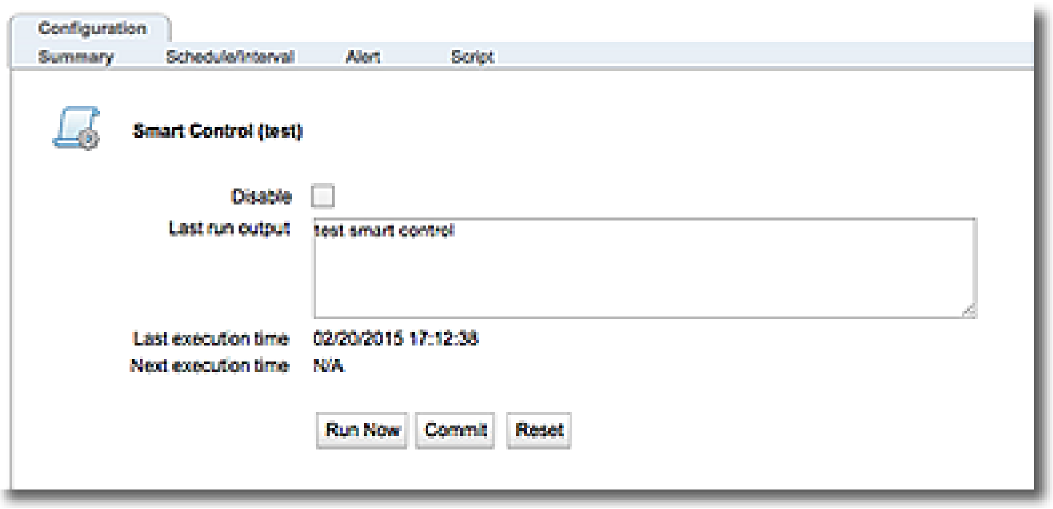
To display the last run output using the CLI enter `eqcli >` **`smart_control`** ***`smartcontrolname`*** **`lastrun`**.

In this example, a simple Smart Control script `echo "test smart control";` is used.

```
eqcli > smart_control test lastrun
Last run output: "test smart control";
```

To display the last run output using the GUI, select **System > Smart Control >** ***Smart Control Name*** on the left navigational pane to display the **Configuration Summary** display on the right. The latest Smart Control run will be displayed in the **Last run output** pane as shown below.

In this example, a simple Smart Control script **`echo "test smart control"`**`;` is used.



Smart Control (test)

Disable

Last run output: test smart control

Last execution time: 02/20/2015 17:12:38

Next execution time: N/A

If Using a Smart Control to Create "eqcollect" Archives

If you run a Smart Control script that executes the “eqcollect” CLI command to create a save state archive (a.k.a. a "collect”), please note the following:

1. It is possible for the process of creating the archive to take longer than the time allotted by the Smart Controls’s “run_limit” parameter. If the "run_limit" parameter is less then the real execution time of the Smart Control, the message: `"79000010: Disabling Smart Control” will be logged in /var/log/eq`. The Smart Control will continue to run until the eqcollect process completes. When it’s done, the archive will be stored in the ADC's */var/crash* directory.
2. If the Smart Control is already running and you attempt to execute it again in either the CLI or the GUI, the following message is displayed: `"eqcli(ERROR): 79000027: This Smart Control is already running. Try again later.”`. You can verify that the Smart Control has completed by checking the */var/log/eq* directory to see if the archive has been generated.

Chapter 24

Alerts

Sections within this chapter include:

Configuring Alerts

An alert is an administratively configured action that is executed whenever an event of a particular type occurs on a particular Equalizer object. For example: a user can be sent an email whenever a particular server is marked UP or DOWN by health check probes.

email, **syslog**, **snmp**,,**ui** , and **smart control** notification types are supported. Multiple notification types can be specified for a single alert.

Alerts can be set up and managed using the GUI or CLI.

Parameters

The **Alert Notification** pane on the **Add Alert** configuration dialogue will change, based on the **Notification Type** selected in the drop down list. For example, the **From**, **To**, and **Subject** fields will be displayed if you select **email** as the **Notification Type**. If you select **smartd** as the **Notification Type**, a **Smart Control** drop down list will be displayed with a list of previously configured **Smart Events**.

Select multiple **Alert Types** and **Notification Types** by pressing the **CTRL** key while selecting each item from the lists.

Alert Configuration Parameters

GUI Parameter (CLI Parameter)	Description
Disable **(disable)**	Disables the alert.
Alert Name **(alert)**	A descriptive name for the alert.
Alert Type	
Alert Type **(alert_type)**	Alert Types are **state_change** and **exception**. The **state_change** alert indicates that the object has transitioned from one state to another (i.e., when a server stops responding to health checks and is marked "down".) An **exception** alert indicates an error condition exists on the object of the alert. (i.e., when all servers in a server pool are marked "down"). **NOTE**: All exception alerts are always logged to the syslog, even if the syslog alert notification type is turned off.
Alert Target	

GUI Parameter (CLI Parameter)	Description
Target Object Name (`object`)	This is the name of the object associated with this alert. In the GUI, the name of the object on which the alert is configured is required. In the CLI, if the object is contained within another object, both the object names must be specified, separated by a colon. For example, to specify a server instance, enter the name of the server pool, a colon (:) and the name of the server instance. **Object Hierarchy** The object hierarchy is a comma-delimited list describing the object hierarchy. An object hierarchy format should be used as follows: **`objname1`**:**`objtype1`**,...**`objname4`**:**`objtype4`**, where: **`objname`*X*** is the name of an object, existing in the configuration, for which the alert definition is to be applied. **`objtype`*X*** is the corresponding type of the object
Target Object Type (`objtype`)	This is the type of object that the alert is being configured on. You can configure alerts on **Failover Groups**, **Interfaces**, **LLB Gateways**, **Peers**, **Servers Server Instances,** and attached **Health Checks**. A peer **state_change** alert is configured on a failover group (**`fogrp`**) object, NOT a peer object. Thus, when a **Failover Group** is used, a **Target Object Name** (**`object`**) and Peer name must be entered. **Note:** A "Primary" flag *can* be set on a failover **Peer** that controls whether any of the alerts for load balancing objects are generated. This is based on the current failover mode of the Failover Group to which the object belongs: If set, alerts for load balancing objects will only be generated for the associated failover groups that are in "Primary" mode. If not set, alerts for load balancing objects will be generated for all failover groups, regardless of mode. If not in failover, this flag has no effect. (Refer to "Configuring Active/Active Failover (CLI)" on page 730.) When **Server Instance** is selected from the drop down list, an additional **Server Pool Name** text field will appear in the GUI where a **Server Pool Name** must be specified.
Alert Notification	

GUI Parameter (CLI Parameter)	Description
Notification Type **(`notify_type`)**	Currently supported are: **email** and **syslog**. You can specify multiple notification types by pressing the **CTRL** key while selecting **notification types**. **email** - will allow you to configure an email alert notification message. Selecting this Notification Type will expand the Alert Notification grouping to include **From**, **To** , and **Subject** fields that are required to send the email notification. **ui** The ui alert notification type notifies users of an alert in the CLI and GUI. **snmp** - SNMP traps send alert notifications via an SMMP trap to send to the currently configured trap servers. Refer to Setting Up SNMP Traps for additional information.. **syslog** - selecting this option sends an alert message to the system log. **smartd** - **smartd** alerts enable notification that a smart event has occurred. If **smartd** is selected from the Notification Type drop down list, a Smart Control drop down list will be visible with configured Smart Controls. You must have a Smart Control configured prior to using this **Notification Type**. Refer to "Adding Smart Events" on page 799 for instructions on adding Smart Controls.

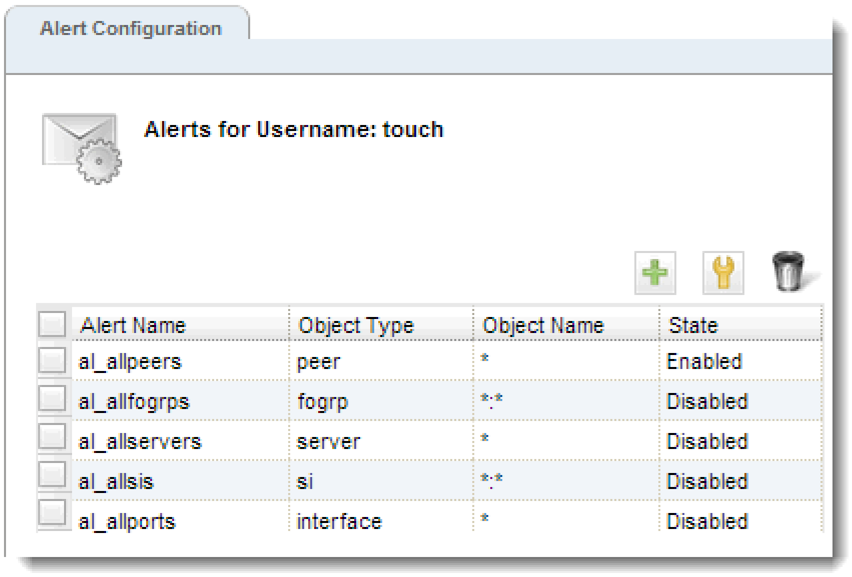
Configuring Alerts in the GUI

Alerts are configured on a per-user basis. A user login name with administrative permissions can specify alerts for any user on any object; users without the administrative permissions can only specify alerts for themselves on objects on which they have permission.

Note - Prior to configuring alerts, you must have previously configured servers, server instances, peers, or user interfaces. In order for a user's alert notification emails to work, a mail server:
1. Must be added to external services.
2. Must be selected in the user's context.

To configure and edit alerts using the GUI:

1. Log in as described in "Logging In" on page 238
2. Select **System > Configuration** on the left navigational pane to display the list of configured alerts on the right as shown below.



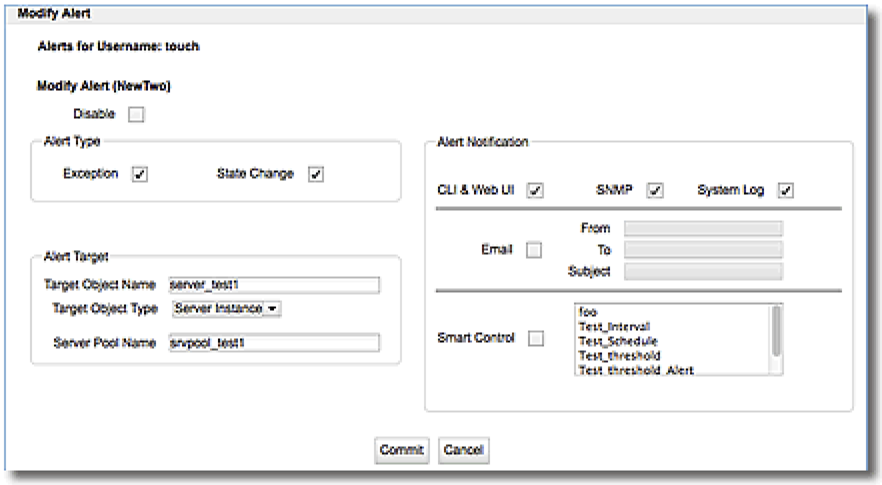
Alerts for Username: touch

	Alert Name	Object Type	Object Name	State
☐	al_allpeers	peer	*	Enabled
☐	al_allfogrps	fogrp	*:*	Disabled
☐	al_allservers	server	*	Disabled
☐	al_allsis	si	*:*	Disabled
☐	al_allports	interface	*	Disabled

3. Click on **+** to display the **Add Alert** configuration dialogue. In the example shown below, an email notification on a server instance has been configured.

   If you would like to edit a previously configured alert, select an alert using the check box and either click on the icon or double click on the selected alert to display the **Modify Alert** dialogue. This dialogue is virtually identical to the **Add Alert** configuration dialogue. The only difference is that you would not be able to enter an **Alert Name** in the space provided.

4. Configure parameters as described in the Alert Configuration Parameters table above. Note that if you change the **Target Object Type** selection from the drop down list, the parameter entries in the **Alert Target** pane will also change, depending on the Target Object Type. Refer to the examples below for details.



Modify Alert

Alerts for Username: touch

Modify Alert (NewTwo)

Disable ☐

Alert Type

Exception ✓ State Change ✓

Alert Target

Target Object Name	server_test1
Target Object Type	Server Instance
Server Pool Name	srvpool_test1

Alert Notification

CLI & Web UI ✓ SNMP ✓ System Log ✓

Email ☐ From To Subject

Smart Control ☐

foo
Test_Interval
Test_Schedule
Test_threshold
Test_threshold_Alert

5. Click on **Commit** to save the Alert configuration. The newly created alert should appear in the list shown in step 4 above.

Use of Wild Cards

When configuring alerts using the GUI, you can use wild card with the **Alert Taget** parameters in the alert configuration. For example;

- If a **Server** is selected as the **Target Object Type.**you can use a wild card character (*) as the **Target Object Name**, this means that you want to configure the alert for *all* Servers.
- If a **Server Instance** is selected as the **Target Object Type** you can use wild card characters (*) as the **Target Object Name** and the **Server Pool Name**, this means that you want to configure the alert on *all* Server Pools and on *all* Server Instances.

Refer to "Using Wild Card Characters to Configure Alerts" on page 829 for additional information on using wild card characters.

Enabling or Disabling Alerts in the GUI

Note - Alerts are enabled by default.

An individual alert can be enabled or disabled as follows:

1. Select **System > Global > Alert Configuration** on the left navigational pane.
2. On the right configuration pane, either double click on the alert that you want to disable or of select the alert and click on the + icon to enter the **Modify Alert** screen.
3. Click on the **Disable** option to disable the alert.

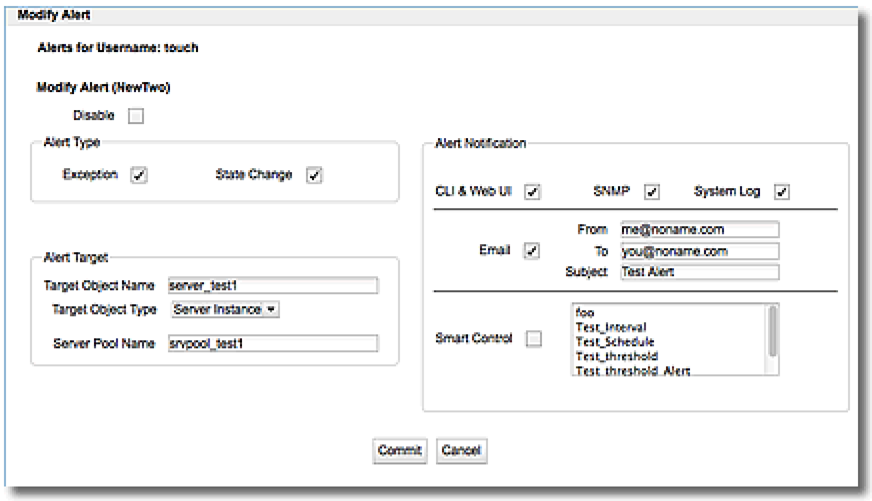
Examples

The following are examples of the configuration of Server Alerts and Peer Alerts using the GUI.

Server Instance Alert Example

Setting an alert on a server instance allows you to send email, log a message to the system log, or both, whenever a server instance is marked up or down by Layer 4 health check probes.

For example, the following alert configuration example creates an alert for the **touch** user that sends email whenever the server instance **TCP Server Instance 1** in server pool named **TCP Server 1** is marked up or down by Layer 4 probes. Note that you will need to enter a **Server Pool Name**. This is the server pool to which the server instance is attached.



Modify Alert

Alerts for Username: touch

Modify Alert (NewTwo)

Disable ☐

Alert Type

Exception ✓ State Change ✓

Alert Target

Target Object Name	server_test1
Target Object Type	Server Instance
Server Pool Name	srvpool_test1

Alert Notification

CLI & Web UI ✓ SNMP ✓ System Log ✓

Email ✓

From	me@noname.com
To	you@noname.com
Subject	Test Alert

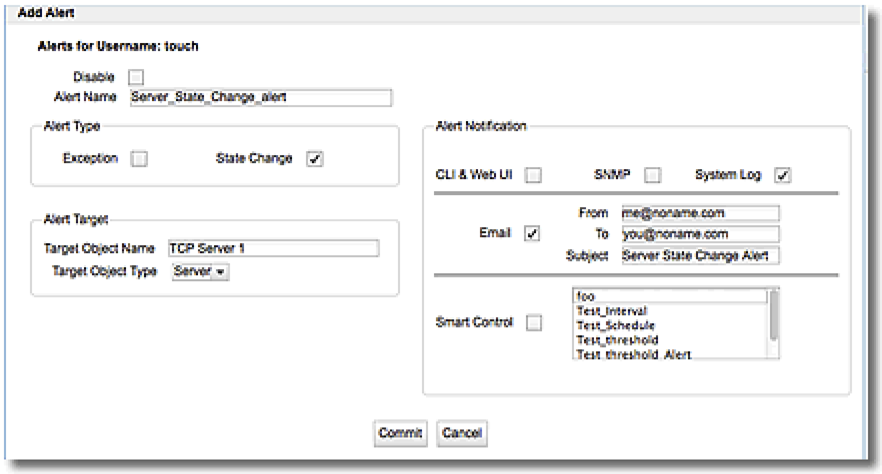
Smart Control ☐

foo
Test_Interval
Test_Schedule
Test_threshold
Test_threshold_Alert

Server Alert Example

Setting an alert on a server allows you to send email, log a message to the system log, or both, whenever a server is marked up or down by Layer 3 health check probes.

For example, the following alert configuration example creates an alert for the **touch** user that sends email whenever the server **TCP Server 1** is marked up or down by Layer 3 probes:



Add Alert

Alerts for Username: touch

Disable ☐

Alert Name: Server_State_Change_alert

Alert Type

Exception ☐ State Change ✓

Alert Target

Target Object Name	TCP Server 1
Target Object Type	Server

Alert Notification

CLI & Web UI ☐ SNMP ☐ System Log ✓

Email ✓

From	me@noname.com
To	you@noname.com
Subject	Server State Change Alert

Smart Control ☐

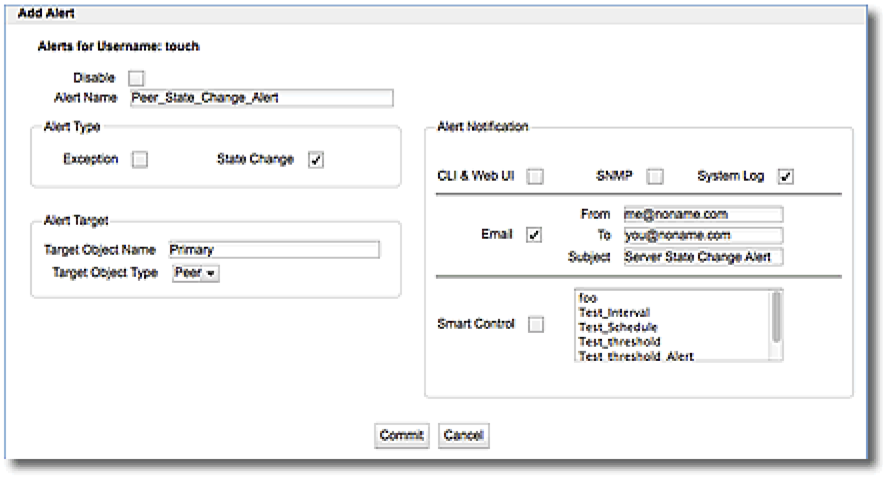
foo
Test_Interval
Test_Schedule
Test_threshold
Test_threshold_Alert

Peer Alert Example

Setting an alert on a peer allows you to send email, log a message to the system log, or both, whenever a peer changes to Primary, Backup, or Standalone modes. Primary and Backup modes apply to Equalizer in a failover configuration. Standalone mode is the normal operational state for a single Equalizer not deployed in a failover pair when it is first booted.

On an Equalizer that is not deployed in a failover configuration, there is a single peer definition that refers to the local Equalizer. In a failover configuration, there are two peer definitions: one for the local Equalizer and one for the remote Equalizer in the failover pair.

For example, the following alert configuration example creates an alert for the **touch** user that sends email whenever the local ADC (peer **Primary**, which is not in failover) reboots and changes to Standalone mode:



Add Alert

Alerts for Username: touch

Disable ☐
Alert Name: Peer_State_Change_Alert

Alert Type
Exception ☐ State Change ✓

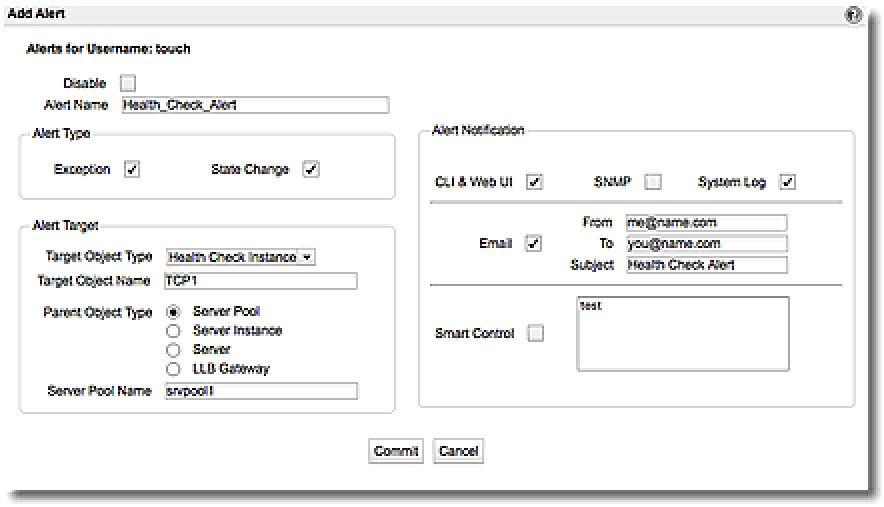
Alert Target
Target Object Name: Primary
Target Object Type: Peer

Alert Notification
CLI & Web UI ☐ SNMP ☐ System Log ✓
Email ✓
From: me@noname.com
To: you@noname.com
Subject: Server State Change Alert
Smart Control ☐
foo
Test_Interval
Test_Schedule
Test_threshold
Test_threshold_Alert

Attached Health Check Alert Example

Setting an alert on an attached health check allows you to send email, log a message to the system log, or both, whenever.an attached health check becomes disabled or the health check parameters have changed (**State Change**) or if an error condition exists (**Exception**)

For example, the following alert configuration example creates an alert that sends email whenever a state change occurs on a health check attached a server instance (**srv1**) on a server pool (**srvpool1**). Note that you will need to select a **Parent Object Type**. This is the object to which the health check is attached.



Add Alert

Alerts for Username: touch

Disable ☐
Alert Name: Health_Check_Alert

Alert Type
Exception ✓ State Change ✓

Alert Target
Target Object Type: Health Check Instance
Target Object Name: TCP1
Parent Object Type: [✓ Server Pool] Server Instance / Server / LLB Gateway
Server Pool Name: srvpool1

Alert Notification
CLI & Web UI ✓ SNMP ☐ System Log ✓
Email ✓ From: me@name.com To: you@name.com Subject: Health Check Alert
Smart Control ☐ test

Configuring Alerts in the CLI

Alerts are configured on a per-user basis. A user with administrative log in credentials can specify alerts for any user on any object; users without administrative credentials can only specify alerts for themselves on objects on which they have permissions.

Note - Prior to configuring alerts, you must have previously configured servers, server instances, peers, or user interfaces. In order for a user's alert notification emails to work, a mail server:
1. Must be added to external services.

2. Must be selected in the user's context.

Refer to "User Permissions" on page 227 for descriptions of object-user permissions.

Prior to configuring alerts, you must have previously configured load balancing objects. In order for a user's alert notification emails to work, a mail server:

- Must be added to external services.
- Must be selected in the user's context.

Use of Wildcards

The last object name in the hierarchy can contain one or more wild card (*) characters. For example, to configure an alert for all server instances of server pool **sp01**, specify:

```
eqcli > object sp01:srvpool,*:si
```

Also, an object hierarchy of **sp01:srvpool,sv*1:si** would configure alerts for server instances: sv1, sv01, sv11, etc.

Refer to "Using Wild Card Characters to Configure Alerts" on page 829 for additional information about the use of wild cards.

Enabling or Disabling Alerts in the CLI

Note - Alerts are enabled by default.

An individual alert can be enabled or disabled by entering:

```
eqcli > user username alert alertname state enable | disable
```

All alerts can be enabled or disabled by entering:

```
eqcli > alerts enable | disable
```

Displaying Alert Notifications

You can display either the list of pending alert notifications (if no other information is on the command line), or details for a specific notification.

An example of displaying all notifications is as follows:

```
eeqcli > show notification
ID  Time Stamp       Type          Obj Type  Obj Name    Alert Name
1   Jan 20 13:57:24  state_change  fogrp     FirstAlert  al_allfogrps
2   Jan 20 13:57:47  state_change  server    servertcp   al_allservers
3   Jan 20 13:57:49  state_change  fogrp     Unassigned  al_allfogrps
```

An example of showing a specific notification by ID is as follow:

```
eqcli > show notification 1
Notification ID       : 1
Generation Timestamp  : Jan 20 13:57:24
Alert Type            : state_change
Alert Subtype         : Backup
Alert Name            : al_allfogrps
Object Type           : fogrp
Object Name           : FirstAlert
Message               : 47000194: Local Peer Tatooine has entered Backup
mode
```

An example of showing the first notification is as follows:

```
eqcli > show notification first
Notification ID       : 1
Generation Timestamp  : Jan 20 13:57:24
Alert Type            : state_change
Alert Subtype         : Backup
Alert Name            : al_allfogrps
Object Type           : fogrp
Object Name           : FirstAlert
Message               : 47000194: Local Peer Tatooine has entered Backup
mode
```

An example of showing the first notification matching one or more filters is as follows. If `object_name` is specified, `object_type` must also be specified.:

```
eqcli > show notification first alert_type state_change object_type fogrp
object_name FirstAlert
Notification ID       : 1
Generation Timestamp  : Jan 20 13:57:24
Alert Type            : state_change
Alert Subtype         : Backup
Alert Name            : al_allfogrps
Object Type           : fogrp
Object Name           : FirstAlert
Message               : 47000194: Local Peer Tatooine has entered Backup
mode
```

Examples

The following are examples of the configuration of Server Alerts and Peer Alerts using the CLI.

Server Instance Alert Example

Setting an alert on a server instance allows you to send email, log a message to the system log, or both, whenever a server instance is marked up or down by Layer 4 health check probes.

For example, the following sequence of commands creates an alert for the user (**`touch`**) that sends email whenever the server (**`testserver`**) in server pool named **`realpool`** is marked up or down by Layer 4 probes:

```
eqcli > user touch
eqcli user-tou*> alert testsrvrinst
eqcli user-tou*-alert-tes*> alert_type state_change
eqcli user-tou*-alert-tes*> notify_type email
eqcli user-tou*-alert-tes*> object realpool:server,testserver:si
eqcli user-tou*-alert-tes*> to user@example.com
eqcli user-tou*-alert-tes*> subject "Server instance status email from
Eq450-100."
eqcli user-tou*-alert-tes*> commit
```

Server Alert Example

Setting an alert on a server allows you to send email, log a message to the system log, or both, whenever a server is marked up or down by Layer 3 health check probes.

For example, the following sequence of commands creates an alert for the user (**touch**) that sends email whenever the server (**testserver**) is marked up or down by Layer 3 probes:

```
eqcli > user touch
eqcli user-tou*> alert testsrvr
eqcli user-tou*-alert-tes*> alert_type state_change
eqcli user-tou*-alert-tes*> notify_type email
eqcli user-tou*-alert-tes*> object testserver:server
eqcli user-tou*-alert-tes*> to user@example.com
eqcli user-tou*-alert-tes*> subject "Server status email from Eq450-100."
eqcli user-tou*-alert-tes*> commit
```

Peer Alert Example

Setting an alert on a peer allows you to send email, log a message to the system log, or both, whenever a peer changes to Primary, Backup, or Standalone modes. Primary and Backup modes apply to Equalizer in a failover configuration. Standalone mode is the normal operational state for a single Equalizer not deployed in a failover pair when it is first booted.

On an Equalizer that is not deployed in a failover configuration, there is a single peer definition that refers to the local Equalizer. In a failover configuration, there are two peer definitions: one for the local Equalizer and one for the remote Equalizer in the failover pair.

For example, the following sequence of commands creates an alert for the user (**touch**) that sends email whenever the local Equalizer (**peer Eq_AD1122CC99**, which is not in failover) reboots and changes to Standalone mode:

```
eqcli > user touch
eqcli user-tou*> alert standmode
eqcli user-tou*-alert-tes*> alert_type state_change
eqcli user-tou*-alert-tes*> notify_type email
eqcli user-tou*-alert-tes*> object Eq_AD1122CC99:peer
eqcli user-tou*-alert-tes*> to user@example.com
eqcli user-tou*-alert-tes*> subject "Status email from Eq450-100."
eqcli user-tou*-alert-tes*> commit
```

Health Check Instance Alert Example

Setting an alert on an attached health check allows you to send email, log a message to the system log, or both, whenever.an attached health check becomes disabled or the health check parameters have changed (state change) or if an error condition exists (exception)

For example, the following alert configuration example creates an alert for the **touch** user that sends email whenever a state change occurs on a health check attached a server instance (`srv1`) on a server pool (`srvpool1`)

It is important to note the syntax when entering `object` details on the health check alert. For example, if you generate an alert on a server instance, you will also need to specify the server pool to which the server instance is attached.

They should be entered using the following syntax:

For a server pool:

```
eqcli user-tou* > object sp_name:srvpool,hc name:hc
```

For a server instance:

```
eqcli user-tou* > object sp_name:srvpool,si_name:si,hc_name:hc
```

Fora server:

```
eqcli user-tou* > object server_name:server,hc_name:hc
```

```
eqcli > user touch
eqcli user-tou*> alert testsrvr
eqcli user-tou*-alert-tes*> alert_type state_change
eqcli user-tou*-alert-tes*> notify_type email
eqcli user-tou*-alert-tes*> object srvpool1:srvpool,srv1:si,TCP1:hc
eqcli user-tou*-alert-tes*> to user@example.com
eqcli user-tou*-alert-tes*> subject "Health check Server Instance Alert"
eqcli user-tou*-alert-tes*> commit
```

Using Wild Card Characters to Configure Alerts

Wild card characters can be used when you want to configure an alert for multiple load balancing objects using the GUI and the CLI. The advantage of this is that you could save time and configure one alert for multiple object types. In alert configuration, an asterisk (*) is used as a wild card character in specific sequences that are described below.

Using Wild Cards with Target Objects

Equalizer's alerts can be configured on the following Target Objects. The table below should be used as a guideline in configuring alerts as these are the specific objects to which alerts can be configured.

Top Level Objects

Top Level Object	Target Object	Configures alerts on...
Servers	`*:server`	Configures alerts on all servers.
Interfaces	`*:interface`	Configures alerts on all interfaces.
Peers	`*:peer`	Configures alerts on all peers.
	`*:*`	Use of the asterisk (*) in this manner will configure an alert for all Top Level objects that includes servers, interfaces, peers, and LLB gateways. (e.g., `*:server, *:interface,` and `*:peer,`)

Second Level Objects

Top Level Object	Target Object	Configures alerts on...
Servers > Health Checks	`*:server,*:hc`	Configures alerts on all servers and all of their attached health checks.
Peers > Failover Groups	`*:peer,*:fogrp`	Configures alerts on all peers and all of their associated failover groups.
LLB-Gateways > Health Checks	`*:llb-gw,*:hc`	Configures alerts on all LLB gateways and all of their attached health checks.
Server Pools > Server Instances	`*:srvpool,*:si`	Configures alerts on all server pools and all of their associated server instances.
Server Pools > Health Checks	`*:srvpool,*:hc`	Configures alerts on all server pools and all of their attached health checks.
	`*:*,*:*`	Use of the asterisk (*) in this manner will configure an alert for all Second Level objects that includes Servers and attached health checks, server pools and their server instances, peers and the failover groups to which they are a part of, and LLB gateways and their attached health checks. (e.g, `*:server,*:hc, *:peer,*:fogrp, *llb-gw,*:hc,*:srvpl,*:si, *:srvpool,*:hc`)

Third Level Objects

Top Level Object	Target Object	Configures alerts on...
Server Pools > Server Instances > Health Checks	`*:srvpool,*:si,*:hc`	Configures alerts on all servers instances on all server pools and all of the health checks attached to server instances.
	`*:*,*:*,*:*`	Use of the asterisk (*) in this manner will configure an alert for Third Level objects that includes server pools, their server instances and health checks attached to the server instances. (e.g., `*:srvpl,*:si,*:hc`).

Basic Alert Configuration using the CLI

When creating an alert in the CLI, the following syntax is used:

```
eqcli > user user name alert alert name object object hierarchy
```

When you specify an object, an object hierarchy is used. This is a comma-delimited list as follows:

```
object hierarchy = <objectname1>:<object type1>,...,<object
name4>:<object type4>,
```

where:

`objectnameX` is the name of an existing object in the configuration, for which the alert configuration is to be applied.

`object typeX` is the corresponding type of load balancing object.

As an example, when you use the object hierarchy.you can set an alert for server instance `sv01`on server pool `sp01` you would specify:

```
eqcli > user myusername alert testalert object sp01:srvpool,sv01:si
```

In `sp01:srvpool,sv01:si` specifies server pool name `sp01` on an object type `srvpool` (server pool)and a server instance `sv01` on an object type `si` (server instance).

Refer to "Configuring Alerts in the CLI" on page 824 for additional alert configuration detail.

Using Wild Cards for Alerts on the CLI

Any object name in the hierarchy can contain one or more wild card (*) characters. For example, to configure an alert for all server instances of all server pools, you would specify:

```
object *:srvpool,*:si
```

Another use of the wild card character is to use it to configure alerts on all object names that use a common character in their names. For example, an object hierarchy of `sp01:srvpool,sv*1:si` would configure alerts for all server instances that use "1" in their names. For example: `sv1, sv01, sv11`, etc.

The following are examples of using wild card characters and the effect that the wild card will have on the alert :

Wildcard Use	Effect

`*:srvpl,*:*`
`*:*,*:si`
	Both examples specify that you want to configure an alert on all server pools, all server instances.

`*:peer,*:*`
`*:*,*:fogrp`
	Both examples specify that you want to configure an alert on all peers, all failover groups.

`*:server,*:*`	This example specifies that you want to configure an alert on all servers and all health checks attached to servers. You could also specify : `*:server,*:hc`.
`*:llb-gw,*:*`	This specifies that you want to configure an alert on all LLB Gateways, all health checks. You could also specify : `*:llb-gw,*:hc` .
`*:*,*:hc`	This specifies that you want to configure an alert on all Second Level objects, all health checks.
`sp01:srvpool,sv*1:si`	This would configure alerts for all server instances that use "1" in their names. For example: `sv1, sv01, sv11`, etc.

`*:srvpl,*:si,*:hc,`
`*:*,*:si,*:hc,`
`*:srvpl,*:*,*:hc,`
`*:srvpl,*:si,*:*,`
`*:*,*:*,*:hc`
	All of these examples mean that you want to configure an alert on all Third level objects, all health checks.

Basic Alert Configuration using the GUI

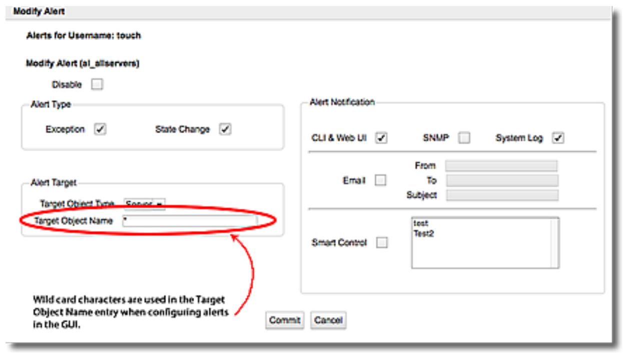
When configuring alerts using the GUI the **System > Global > Alert Configuration** screen is used. The **Target Object Type** drop down list is used to select a **Target Object Type**. The name of the **Target Object** is entered in the **Target Object Name** entry box. When you select a **Server Instance** as the **Target Object Type**, a **Server Pool Name** entry box is displayed where the name of the Server Pool, to which the server instance is entered.

Refer to "Configuring Alerts in the GUI" on page 819 for additional alert configuration detail.

Using Wild Cards when Configuring Alerts in the GUI

The wild card character can be used with the **Target Object Name** and also with the **Server Pool** name if **Server Instance** is selected as the **Target Object**. In the examples below:

- **Server** is selected as the **Target Object Type**. If you use a wild card character (*) as the **Target Object Name**, this means that you want to configure the alert for *all* Servers.
- **Server Instance** is selected as the **Target Object Type**. If you use wild card characters (*) as the **Target Object Name** and the **Server Pool Name**, this means that you want to configure the alert on *all* Server Pools and on *all* Server Instances.


Modify Alert

Alerts for Username: touch

Modify Alert (al_allservers)

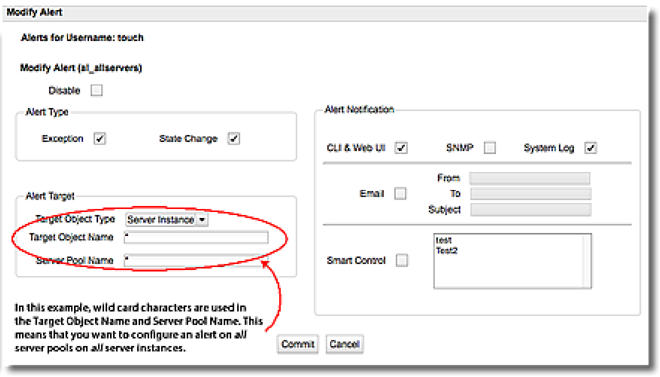
Disable [ ]

Alert Type: Exception [✓] State Change [✓]

Alert Target: Target Object Type: Server; Target Object Name: *

Alert Notification: CLI & Web UI [✓] SNMP [ ] System Log [✓]; Email [ ] From / To / Subject; Smart Control [ ] test Test2

Wild card characters are used in the Target Object Name entry when configuring alerts in the GUI.


Modify Alert

Alerts for Username: touch

Modify Alert (al_allservers)

Disable [ ]

Alert Type: Exception [✓] State Change [✓]

Alert Target: Target Object Type: Server Instance; Target Object Name: *; Server Pool Name: *

Alert Notification: CLI & Web UI [✓] SNMP [ ] System Log [✓]; Email [ ] From / To / Subject; Smart Control [ ] test Test2

In this example, wild card characters are used in the Target Object Name and Server Pool Name. This means that you want to configure an alert on *all* server pools on *all* server instances.

Configuring an SMTP Relay

Email alerts require a configured SMTP relay in order to send email to the recipient specified in the alert definition. To set up an SMTP relay, you need to know:

- The SMTP server’s IP address or Fully Qualified Domain Name (FQDN). If an FQDN is used, DNS must also be configured.
- The port on which the SMTP server accepts incoming mail (usually port 25).

Note - The first SMTP Relay object set as the default in the configuration. If a user configures an email alert and does not configure an SMTP relay, the default will be used.

Configuring an SMTP Relay Using the CLI

Currently, Equalizer supports one SMTP relay. The format of the global CLI command to create a new SMTP relay is:

```
eqcli > ext_services smtp_relay name server IP_or_FQDN port number
```

For example, if you have an SMTP relay server named **postmaster** that has an IP address of 10.0.0.111 and uses the standard SMTP port, you can enter this command:

```
eqcli > ext_services smtp_relay postmaster server 10.0.0.111 port 25
```

To display the SMTP relay definition, enter:

```
eqcli > show ext_services smtp_relay postmaster
```

To delete the SMTP relay definition, enter:

```
eqcli > no ext_services smtp_relay postmaster
```

To modify an existing SMTP relay definition, specify new values for the desired parameters. For example, this command changes the IP address for **postmaster**:

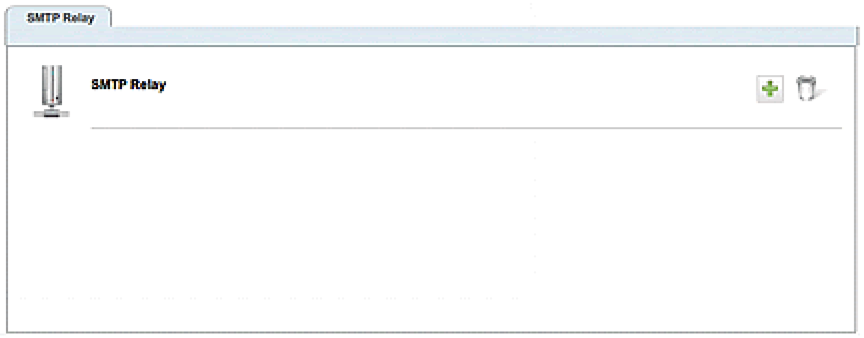
```
eqcli > ext_services smtp_relay postmaster server 172.16.0.123
```

Configuring an SMTP Relay Using the GUI

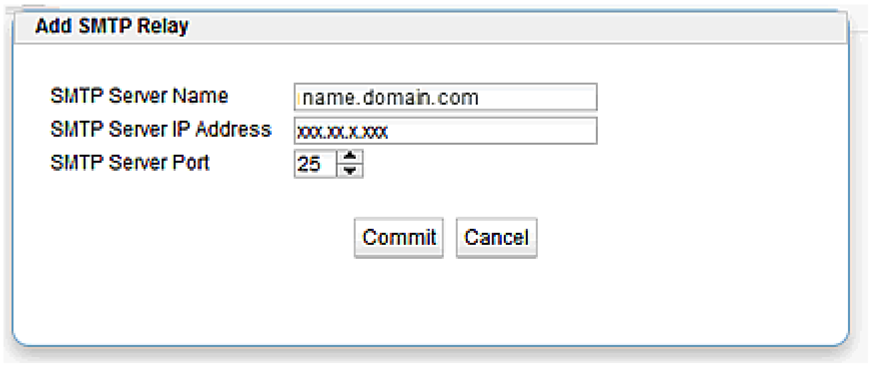
SMTP Relays are commonly used when you want to configure email alerts. With email alerts, you be adding email addresses to the alert.

The **SMTP Relay** screen is used to specify an SMTP Relay Server and specify an IP address and Equalizer port to use. It is accessed by clicking on the host name on the left navigational pane and selecting the **External Services** tab.

Currently, only one SMTP relay is supported.



To add and SMTP relay, click on ➕ to display the **Add SMTP Relay** form as shown below:



Add SMTP Relay

SMTP Server Name	name.domain.com
SMTP Server IP Address	xxx.xx.x.xxx
SMTP Server Port	25

Enter an IP Address for the SMTP Relay in the **SMTP Server IP Address** field. Specify an Equalizer port to use using the **SMTP Server Port** selection. The **Port** defaults to **25** and can range from 1 to 65535.

Click on **Commit** to save the entries.

To delete an STMP relay, select the relay and click on the 🗑 icon.

Chapter 25

Using SNMP Traps

Sections within this chapter include:

Setting Up SNMP Traps

SNMP traps are alerts that are tied into the Equalizer Alerts system. They enable an agent to notify a management station of significant events by way of unsolicited SNMP messages. First, they must be enabled using the CLI context and then created for each desired alerts. Presently, Equalizer supports the following SNMP traps:

- Server up/down events - Equalizer will triggers these traps when it detects either a server failure or a response from a failed server.
- Peer state change events - Equalizer will trigger these traps whenever a state change occurs to a peer or peer interface..
- Failover Group state change events - Equalizer triggers these traps whenever a state change occurs to a failover group.

The Simple Network Management Protocol (SNMP) is an internet standard that allows a management station to monitor the status of a device over the network. SNMP organizes information about Equalizer and provides a standard way to help gather that information. Using SNMP requires:

- An SNMP agent running on the system to be monitored.
- A Management Information Base (MIB) database on the system to be monitored.
- An SNMP management station running on the same or another system.

A management station is not provided with Equalizer and must be obtained from a third party supplier. The management station is often used primarily to browse through the MIB tree, and so is sometimes called a MIB browser. One such management station that is available in a free personal edition is the *iReasoning MIB Browser*, available from www.ireasoning.com.

A MIB database is a hierarchical tree of variables whose values describe the state of the monitored device. A management station that wants to browse the MIB database on a device sends a request to the SNMP agent running on the device. The agent queries the MIB database for the variables requested by the management station, and then sends a reply to the management station.

When configuring SNMP traps, an alert type can ONLY be "state change" and the object type can ONLY be those above. In this context, "state change" is when an object transitions from "UP" to "DOWN" or vice-versa.

Different OIDs (Object Identifier) are used for each item in the above list. The OID identifies a variable that can be read via SNMP. So, for example, server up and server down events are sent using different OIDs. The text message sent with the trap tells you exactly what the event was (server up or server down).

The SNMP Trap configuration process includes 4 steps:

I. Set up an SNMP Management Station
II. Enable SNMP

III. Enable SNMP Traps
IV. Creating alerts for the desired traps

Setting Up an SNMP Management Station

An SNMP management station is not provided with Equalizer. In order to use SNMP to manage Equalizer, a third-party management console must be installed and configured on a machine that can access Equalizer. Configuration procedures are specific to the management console used.

At a minimum, the SNMP management console needs to be configured to:

- Use Equalizer's IP address and port 161 for SNMP requests.
- Use the community string specified for Equalizer (Refer to "SNMP Commands" on page 219 or "SNMP" on page 278).
- Use the address and port specified in the above for SNMP traps (usually port 162 is used for this purpose, but this can be configured as shown in "Enabling SNMP Traps" on page 841).
- Use the MIB definitions; these need to be loaded into the management console, following the instructions for the console. The MIB source files are located at:

  http://<*Load balancer*-ip>/eqmanual/CPS-EQUALIZER-v10-MIB.my

  http://<*Load balancer*-ip>/eqmanual/CPS-REGISTRATIONS-v10-MIB

  In the above, **<Equalizer-ip>** is the IP address of the Equalizer.

Enabling SNMP

By default, SNMP is a globally enabled service -- meaning that it will run on any subnet that is configured to offer the SNMP service. You must specifically enable SNMP on the subnet or subnets on which you want it to listen for SNMP MIB browser and management station connections.

SNMP can be enabled on *at most* one IPv4 subnet address/port and one IPv6 subnet address/port. SNMP runs on Equalizer's IP address on the configured subnet. Currently, SNMP runs on the default SNMP port (161) only.

To enable SNMP, you must enable it at the global level, and then enable it on any single IPv4 subnet, any single IPv6 subnet, or both. For this procedure, we assume the existence of a properly configured VLAN (172net) and its Default subnet.

1. In the global CLI context, confirm that SNMP is globally enabled:

```
   eqcli > show
```

   You should see a line that looks like the following:

```
eqcli > show

Variable                Value
recv_timeout            2
conn_timeout            1
hb_interval             2
retry_interval          5
strike_count            3
icmp_interval           15
icmp_maxtries           3
hostname                Equalizer
date                    Thu Mar 20 11:49:09 UTC 2014
timezone                UTC
locale                  en
global services         http, https, ssh, fo_snmp, snmp, Envoy, Envoy_agent
name-servers            None
ntp-server              pool.ntp.org - Unavailable: name-server undefined
syslog-server           None
GUI logo                Coyote Point Systems, A Subsidiary of Fortinet, Inc..
boot image              Equalizer Image B  EQ/OS 10.3.2d (Build XXXXX)

eqcli >
```

On the `global services` line, the `snmp` service means that SNMP is globally enabled for any Equalizer subnet IP address. “`fo_snmp`” means that SNMP is globally enabled for any subnet failover IP address. If either of these keywords has a preceding exclamation point (**!**), then SNMP is disabled for that class of IP addresses. You can enable and disable these flags using the services command, as shown in "Global Commands" on page 164.

2. Now, enable SNMP on the desired VLAN subnet, on either the subnet IP address or the subnet failover (aka “virtual”) IP address. In this example, we enable it on the subnet IP address:

```
eqcli > vlan 172net subnet Default services snmp
```

SNMP is now enabled and will respond to MIB browser requests received on the **172net:Default** subnet IP address (port 161).

Before accessing Equalizer via SNMP, download and install the Equalizer MIB files into your MIB browser, as explained in the following section.

Refer to "SNMP" on page 278 for details on setting SNMP parameters using the GUI.

Enabling SNMP Traps

SNMP traps must first be enabled using the CLI. An snmp trap address and port is required to enable the traps. Enter the following at the CLI prompt:

```
eqcli> snmp serverip ip serverport port
```

where: **`<ip>`** is the SNMP trap server IP and ***`port`*** is the SNMP trap server port.

The port is optional. If it is NOT entered, the default trap server port (162) will be used.

Multiple trap servers can be defined if desired. If they are, ALL traps are sent to ALL configured IPs. The **`show snmp`** command displays the configured trap servers.

A trap server IP can be deleted by entering:

```
eqcli > no snmp serverip
```

All trap server IPs may be deleted by entering:

```
eqcli > no snmp serverip
```

Creating SNMP Trap Alerts

SNMP Traps are configured as alerts and are configured on a per-user basis. A user login name with the admin flag can specify alerts for any user on any object; users without the admin flag can only specify alerts for themselves on objects on which they have permission.

Creating SNMP Trap Server Alerts

Setting an SNMP Trap server alert enables the sending of snmp trap messages to snmp management stations whenever a server is marked up or down by a health check.

For example, the following sequence of commands creates an alert for the **touch** user that sends an snmp trap message whenever the server **testserver** is marked up or down by a health check:

```
eqcli > user touch
eqcli user-tou*> alert testsrvr
eqcli user-tou*-alert-tes*> alert_type state_change
eqcli user-tou*-alert-tes*> notify_type snmp
eqcli user-tou*-alert-tes*> object testserver
eqcli user-tou*-alert-tes*> object_type server
eqcli user-tou*-alert-tes*> commit
```

Creating SNMP Trap Peer Alerts

Setting an SNMP Trap alert enables the sending of snmp trap messages to the snmp management station whenever a peer state changes to Primary, Backup, or Standalone modes. Primary and Backup modes apply to Equalizer in a failover configuration. Standalone mode is the normal operational state for a single Equalizer not deployed in a failover pair when it is first booted.

For example, the following sequence of commands creates an snmp trap alert for the **touch** user that enables trap messages whenever the local Equalizer (peer Eq_AD1122CC99, which is not in failover) reboots and changes to Standalone mode:

```
eqcli > user touch
eqcli user-tou*> alert standalonemode
eqcli user-tou*-alert-tes*> alert_type state_change
eqcli user-tou*-alert-tes*> notify_type snmp
eqcli user-tou*-alert-tes*> object Eq_AD1122CC99
eqcli user-tou*-alert-tes*> object_type peer
eqcli user-tou*-alert-tes*> commit
```

Creating SNMP Trap Failover Group Alerts

Setting an SNMP Trap alert enables the sending of snmp trap messages to the snmp management station whenever a Failover Group changes to Primary, Backup, or Standalone modes.

For example, the following sequence of commands creates an snmp trap alert for the **touch** user that enables a trap message whenever a failover group (`fo_group1`) changes state. Note that the failover group object name must be entered as **`<peername>:<fogroupname>`**:

```
eqcli > user touch
eqcli user-tou*> alert fogroupstatechange
eqcli user-tou*-alert-tes*> alert_type state_change
eqcli user-tou*-alert-tes*> notify_type snmp
eqcli user-tou*-alert-tes*> object Eq_AD1122CC99:fo_group1
eqcli user-tou*-alert-tes*> object_type fogrp
eqcli user-tou*-alert-tes*> commit
```

Chapter 26

How to Use Regular Expressions

Sections within this chapter include:

Regular Expression Terms

The terms in this section describe the components of regular expressions.

- A regular expression (RE) is one or more non-empty branches, separated by pipe symbols (`|`). An expression matches anything that matches one of the branches.
- A branch consists of one or more concatenated pieces. A branch matches a match for the first piece, followed by a match for the second, and so on.
- A piece is an atom optionally followed by a single *, +, or ?, or by a bound.
  - An atom followed by an asterisk (*) matches a sequence of 0 or more matches of the atom.
  - An atom followed by a plus sign (+) matches a sequence of 1 or more matches of the atom.
  - An atom followed by a question mark (?) matches a sequence of 0 or 1 matches of the atom.
- A bound consists of an open brace ({) followed by an unsigned decimal integer, between 0 and 255 inclusive. You can follow the first unsigned decimal integer with a comma, or a comma and a second unsigned decimal integer. Close the bound with a close brace (}). If there are two integers, the value of the first may not exceed the value of the second.

Learning About Atoms

An atom followed by a bound that contains one integer i and no comma matches a sequence of exactly i matches of the atom. An atom followed by a bound that contains one integer i and a comma matches a sequence of i or more matches of the atom. An atom followed by a bound containing two integers i and j matches a sequence of i through j (inclusive) matches of the atom. An atom can consist of any of the following:

- A regular expression enclosed in parentheses, which matches a match for the regular expression.
- An empty set of parentheses, which matches the null string.
- A bracket expression.
- A period (.), which matches any single character.
- A carat (^), which matches the null string at the beginning of a line.
- A dollar sign ($), which matches the null string at the end of a line.
- A backslash (\) followed by one of the following characters: ^.[$()|*+?{\, which matches that character taken as an ordinary character.
- A backslash (\) followed by any other character, which matches that character taken as an ordinary character (as if the \ had not been present).
- A single character with no other significance, which simply matches that character. **Note that regular expressions are case-insensitive.**
- An open brace ({) followed by a character other than a digit is an ordinary character, not the beginning of a bound. It is illegal to end a real expression with a backslash (\).

Creating a Bracket Expression

A bracket expression is a list of characters enclosed in brackets ([...]). It normally matches any single character from the list. If the list begins with ^, it matches any single character not from the rest of the list. Two characters in a list that are separated by '-' indicates the full range of characters between those two (inclusive) in the collating sequence; for example, '[0-9]' in ASCII matches any decimal digit. It is illegal for two ranges to share an endpoint; for example, 'a-c-e'. Ranges are very collating-sequence-dependent, and portable programs should avoid relying on them.

- To include a literal ']' in the list, make it the first character (following an optional '^').
- To include a literal '-', make it the first or last character, or the second endpoint of a range.
- To use a literal '-' as the first endpoint of a range, enclose it in '[.' and '.]' to make it a collating element (see below).

With the exception of these and some combinations using '[' (see next paragraphs), all other special characters, including '\', lose their special significance within a bracket expression.

Within a bracket expression, a collating element (a character, a multi-character sequence that collates as if it were a single character, or a collating-sequence name for either) enclosed in '[.' and '.]' stands for the sequence of characters of that collating element. The sequence is a single element of the bracket expression's list. A bracket expression containing a multi-character collating element can thus match more than one character; e.g., if the collating sequence includes a 'ch' collating element, then the real expression '[[.ch.]]*c' matches the first five characters of 'chchcc'.

Within a bracket expression, a collating element enclosed in '[' and `]' is an equivalence class, representing the sequences of characters of all collating elements equivalent to that one, including itself. (If there are no other equivalent collating elements, the treatment is as if the enclosing delimiters were '[.' and '.]'.) For example, if 'x' and 'y' are the members of an equivalence class, then '[[x]]', '[[y]]', and '[xy]' are all synonymous. An equivalence class may not be an end-point of a range.

Within a bracket expression, the name of a character class enclosed in '[:' and ':]' stands for the list of all characters belonging to that class.

There are two special cases of bracket expressions: the bracket expressions '[[:<:]]' and '[[:>:]] ' match the null string at the beginning and end of a word respectively. A word is defined as a sequence of word characters that is neither preceded nor followed by word characters. A word character is an alnum character (as defined by ctype(3)) or an underscore. This is an extension, compatible with but not specified by IEEE Std 1003.2 (''POSIX.2''), and should be used with caution in software intended to be portable to other systems.

Escape Sequences

The following escape character sequences match the indicated characters:

\	matches a single backslash (\)
\b	matches the beginning of a word (e.g.: `\bex` matches "`example`" but not "`text`")
\n, \r, \t, \v	match whitespace characters
\', \"	match single and double quotes

Matching in Regular Expressions

If a real expression could match more than one substring of a given string, the real expression matches the one starting earliest in the string. If the real expression could match more than one substring starting at that point, it matches the longest. Subexpressions also match the longest possible substrings, subject to the constraint that the whole match be as long as possible, with subexpressions starting earlier in the real expression taking priority over ones starting later. Note that higher-level subexpressions thus take priority over their lower-level component subexpressions.

Match lengths are measured in characters, not collating elements. A null string is considered longer than no match at all. For example, 'bb*' matches the three middle characters of 'abbbc', '(wee|week)(knights|nights)' matches all ten characters of 'weeknights', when '(.*).*' is matched against 'abc' the parenthesized subexpression matches all three characters, and when '(a*)*' is matched against 'bc' both the whole real expression and the parenthesized subexpression match the null string.

Using Regular Expressions in Responders

In some cases, it may be desirable to examine the URL of an incoming request and re-use parts of it in the URL returned to the client by a Redirect Responder. This is the purpose of the regex parameter: specify a custom regular expression that is used to:

- Parse the URL of an incoming request
- Break it down into separate strings (based on the positions of literal characters in the expression)
- Assign each string to a named variable

These named variables can then be used in the URL field of the Redirect Responder. When the Responder replies to a client, it performs string substitution on the URL. Because the purpose of using regular expressions to perform string substitution in Redirect URLs is to parse request URLs into strings, constructing an appropriate regular expression requires an exact knowledge of the format of the request URLs that will typically be coming in to the cluster IP.

Chapter 27 Troubleshooting

Sections within this chapter include:

Connectivity and Configuration Issues

Many connectivity and configuration issues can be diagnosed using standard network troubleshooting techniques. This section identifies some common problems, the most likely causes, and the best solutions. It also describes the diagnostic commands available in the CLI (See "Context Command Summaries" on page 163)

No Output (or Garbled Output) Over Serial

Terminal or terminal emulator may not be properly configured

Check the serial cable connection and the communication settings of the terminal or terminal emulator. Refer to "Setting Up a Terminal or Terminal Emulator" on page 46 for the appropriate settings for your appliance. If you are using terminal emulation software on a Windows or Unix system, make sure the terminal emulation software is connecting to the port to which the serial cable is connected.

Clients Time Out Trying to Contact a Virtual Cluster

Packets from the server are not being routed back through ADC

Log on to the server(s) and check the routing tables. Perform a `traceroute` (or `tracert` on Windows) from the server to the client. Adjust route until the ADCs address shows up in the traceroute output.

All packets sent from the server back to clients must pass through the ADC on the way back to the client unless the spoof cluster option is disabled, or Direct Server Return (DSR) is configured.

Test client is on the same network as the servers

If the test client is on the same network as the servers, the servers will probably try to send data packets directly to the client, bypassing the ADC. You can correct this by adding host routes on the servers so that the servers send their reply packets back to the client through the ADC.

No active servers in the virtual cluster

Possible solutions:

- Check the cluster configuration: Is a server pool assigned to the cluster? Are there server instances in the server pool and are they all marked UP?
- Log onto one of the servers and run the netstat command. If the netstat output shows connections in the SYN-RCVD state, the server is not forwarding its reply packets.

The ADC is not active

Try to ping one of the configured subnet IP addresses. If you do not get a response, “Equalizer Doesn’t Respond to Pings to the Admin Address” provides additional troubleshooting information.

Backup Equalizer Continuously Reboots

Primary and Backup Equalizer Are in a Conflict over Primary

Certain Dell and Cisco switches have Spanning Tree enabled by default. This can cause a delay in the times that the network is accessible and cause the backup unit to enter into failover mode. If you cannot disable Spanning Tree, enable PortFast for all ports connected to the ADCs.

Can’t View the GUI

GUI not enabled

Services flags at the global and subnet level control availability of the GUI. It must be enabled globally and on at least one subnet in order to be accessible. Access the GUI using the subnet IP address on which either of the HTTP or HTTPS services is enabled.

ADC is not active

Try to `ping` the administration address. If you do not get a response, see “Equalizer doesn’t respond to pings to the admin address” below, which provides additional troubleshooting information.

Browser cache needs to be cleared

After an upgrade, the GUI may need to be completely reloaded before it can be used again, depending on the level of change to the GUI code contained in the upgrade. You can do this by clearing your browser cache; or, closing your browser and opening it again to establish a new connection.

Equalizer Doesn’t Respond to Pings to the Admin Address

Equalizer is not powered on

Check that power switch is on and the front panel LED is lit. Connect the keyboard and monitor, cycle the power, and watch the startup diagnostic messages.

Equalizer isn't connected to your network

Check the network wiring.

Use diagnostic tools to check connectivity and routing

See "Using Diagnostic Commands" on page 857 for a list of the diagnostic tools available in the CLI.

Packets from the Server Aren’t Routed Correctly

IP spoofing is enabled

This problem normally occurs in a single network setup. When you enable IP spoofing, clustered servers see the client’s IP address. If the server tries to reply directly to the client, the client will reject the reply (it had sent its request to a different address).

Run a `traceroute` to ensure that routes from a server to a client go through Equalizer and not directly back to the client. If Equalizer does not appear, modify the route to include Equalizer. Alternatively, you can disable IP spoofing.

Web Server Cannot Tell Whether Incoming Requests Originate Externally or Internally

IP Spoofing is not enabled

Check the cluster's configuration and enable the spoof option – this will cause the client's IP address to be used as the source address in packets sent to the server. Also ensure that responses from the server go through Equalizer.

Why aren't my clusters working if the server status is "up"?

There are several reasons this could be happening. Make sure that Equalizer is being used as the default gateway on all your servers, and that the server service or daemon is running. Sometimes additional host or network routes will need to be added to the clustered servers in single network. The `traceroute` (Unix) and `tracert` (Windows) commands are useful diagnostic tools. Trace from the clustered server back to any client that is not able to resolve the cluster address. If Equalizeris not showing up as the first hop, routing is the cause of the problem.

Context Help Does Not Appear

Turn off the Pop-up Blocker for your browser.

Log Contains SSL Errors with "wrong version number"

If you have one or more HTTPS clusters defined, you may see the following messages in the Equalizer log:

```
   ssl_err: 425:error:1408F10B: SSL routines:SSL3_GET_RECORD:wrong version number:s3_
pkt.c:360: ssl_err: fatal error with ip_address
```

These messages indicate that a client has sent an HTTPS request to an HTTPS cluster, but has requested an SSL/TLS version that is not configured on the cluster. These messages are logged by the SSL implementation used by Equalizer and do not necessarily indicate a problem on Equalizer.

For example, if you configure the HTTPS cluster to support SSLv3 ciphers only, then any time a client requests a connection using an SSLv2 cipher, SSL will log these messages. Check the cipher suite for the HTTPS cluster and the configuration of the client to ensure that the desired SSL versions are being used.

Using Diagnostic Commands

Diagnostic commands using the CLI are available to allow an administrator to view information such as:

- Network ARP statistics
- DNS look up
- Network status information
- The top processes of the system
- Analyze packet activity using tcpdump.)
- Disk space on the appliance file system
- The state of the appliance interfaces (ports)
- System processes
- Save descriptions of top processes on the network

Access these commands using h the CLI either in a global context or in `diags` context. For example:

```
eqcli diags > arp | df | dig | ifconfig | netstat | ps | top
```

arp

The `arp` command display the ARP entries. An example of an `arp` output is as follows:

```
eqcli diags> arp
ARP Entries:
? (172.16.0.1) at 00:01:30:b9:69:90 on ixg0
? (192.168.0.15) at (incomplete) on ixg1
? (192.168.0.16) at (incomplete) on ixg1
? (192.168.0.17) at (incomplete) on ixg1
? (192.168.0.18) at (incomplete) on ixg1
? (192.168.0.50) at (incomplete) on ixg1
? (192.168.0.51) at (incomplete) on ixg1
? (192.168.0.52) at (incomplete) on ixg1
? (192.168.0.53) at (incomplete) on ixg1
eqcli diags>
```

df

The `df` command displays the disk space and file system on your appliance. An examle of a `df` output is as follows:

```
eqcli diags> df
Df output:
Filesystem          1K-blocks        Used       Avail %Cap Mounted on
/dev/wd0b             1032238      152378      828250  15% /
/dev/wd0g             1032238        3698      976930   0% /etc
/dev/wd0l             1032238      197626      783002  20% /var
/dev/wd0k             1032238      172354      808274  17% /var/eq
kernfs                      1           1           0 100% /kern
ptyfs                       1           1           0 100% /dev/pts
tmpfs                    8192          16        8176   0% /tmp
tmpfs                   12288         408       11880   3% /tmp/statcache
/dev/wd0h            82575976    60033804    18413374  76% /var/crash
/usr/local/captive    1032238      152378      828250  15%
/tmp/capt.0000516aa
/etc/resolv.conf      1032238        3698      976930   0%
/tmp/capt.0000516aa/etc/resolv.conf
/dev/clockctl         1032238      152378      828250  15%
/tmp/capt.0000516aa/dev/clockctl
tmpfs                     256           4         252   1%
/tmp/capt.0000516aa/var
eqcli diags>
```

dig

The `dig` command displays the DNS lookup information.

```
eqcli diags> dig
Dig output:
; <<>> DiG 9.9.1-P4 <<>>
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36827
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 23
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;.                              IN      NS
;; ANSWER SECTION:
.                       243790  IN      NS      a.root-servers.net.
.                       243790  IN      NS      m.root-servers.net.
.                       243790  IN      NS      f.root-servers.net.
.                       243790  IN      NS      e.root-servers.net.
.                       243790  IN      NS      c.root-servers.net.
.                       243790  IN      NS      k.root-servers.net.
.                       243790  IN      NS      d.root-servers.net.
.                       243790  IN      NS      h.root-servers.net.
.                       243790  IN      NS      b.root-servers.net.
.                       243790  IN      NS      j.root-servers.net.
.                       243790  IN      NS      l.root-servers.net.
.                       243790  IN      NS      i.root-servers.net.
.                       243790  IN      NS      g.root-servers.net.
;; ADDITIONAL SECTION:
h.root-servers.net.     330190  IN      A       128.63.2.53
h.root-servers.net.     330190  IN      AAAA    2001:500:1::803f:235
g.root-servers.net.     330190  IN      A       192.112.36.4
b.root-servers.net.     330190  IN      A       192.228.79.201
c.root-servers.net.     330190  IN      A       192.33.4.12
m.root-servers.net.     330190  IN      A       202.12.27.33
m.root-servers.net.     330190  IN      AAAA    2001:dc3::35
f.root-servers.net.     330190  IN      A       192.5.5.241
f.root-servers.net.     330190  IN      AAAA    2001:500:2f::f
e.root-servers.net.     330190  IN      A       192.203.230.10
k.root-servers.net.     330190  IN      A       193.0.14.129
k.root-servers.net.     330190  IN      AAAA    2001:7fd::1
j.root-servers.net.     330190  IN      A       192.58.128.30
j.root-servers.net.     330190  IN      AAAA    2001:503:c27::2:30
l.root-servers.net.     330190  IN      A       199.7.83.42
l.root-servers.net.     330190  IN      AAAA    2001:500:3::42
d.root-servers.net.     330190  IN      A       199.7.91.13
d.root-servers.net.     330190  IN      AAAA    2001:500:2d::d
i.root-servers.net.     330190  IN      A       192.36.148.17
i.root-servers.net.     330190  IN      AAAA    2001:7fe::53
a.root-servers.net.     330190  IN      A       198.41.0.4
a.root-servers.net.     330190  IN      AAAA    2001:503:ba3e::2:30
```

```
;; Query time: 34 msec
;; SERVER: 10.0.0.123#53(10.0.0.123)
;; WHEN: Wed Mar  5 19:59:18 2014
;; MSG SIZE  rcvd: 699
eqcli diags>
```

ifconfig

The `ifconfig` command displays the state of all interfaces.An example of an `ifconfig` output is as follows:

```
eqcli diags> ifconfig
Ifconfig output:
ixg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 4839
capabilities=3ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_
Tx,UDP4CSUM_Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_Tx>
enabled=3ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_Tx,UDP4CSUM_
Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_Tx>
address: 00:0c:bd:05:a3:04
media: Ethernet autoselect (10GbaseSR full-duplex)
status: active
inet 172.16.5.90 netmask 0xfffff800 broadcast 172.16.7.255
inet alias 172.16.5.93 netmask 0xffffffff broadcast 172.16.5.93
inet alias 172.16.5.95 netmask 0xffffffff broadcast 172.16.5.95
inet alias 172.16.5.96 netmask 0xffffffff broadcast 172.16.5.96
inet alias 172.16.5.91 netmask 0xffffffff broadcast 172.16.5.91
inet alias 172.16.5.97 netmask 0xffffffff broadcast 172.16.5.97
inet alias 172.16.5.92 netmask 0xffffffff broadcast 172.16.5.92
inet alias 172.16.5.94 netmask 0xffffffff broadcast 172.16.5.94
inet6 fe80::20c:bdff:fe05:a304%ixg0 prefixlen 64 scopeid 0x1
ixg1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 4839
capabilities=3ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_
Tx,UDP4CSUM_Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_Tx>
enabled=3ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_Tx,UDP4CSUM_
Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_Tx>
address: 00:0c:bd:05:a3:05
media: Ethernet autoselect (10GbaseSR full-duplex)
status: active
inet 192.168.5.90 netmask 0xfffff800 broadcast 192.168.7.255
inet6 fe80::20c:bdff:fe05:a305%ixg1 prefixlen 64 scopeid 0x2
wm0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 9000
capabilities=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_
Tx,UDP4CSUM_Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_
Tx,TSO6>
enabled=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_Tx,UDP4CSUM_
Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_Tx,TSO6>
address: 00:0c:bd:05:a2:fc
```

```
media: Ethernet autoselect (none)
status: no carrier
inet6 fe80::20c:bdff:fe05:a2fc%wm0 prefixlen 64 scopeid 0x3
wm1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 9000
capabilities=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_
Tx,UDP4CSUM_Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_
Tx,TSO6>
enabled=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_Tx,UDP4CSUM_
Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_Tx,TSO6>
address: 00:0c:bd:05:a2:fd
media: Ethernet autoselect (none)
status: no carrier
inet6 fe80::20c:bdff:fe05:a2fd%wm1 prefixlen 64 scopeid 0x4
wm2: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 9000
capabilities=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_
Tx,UDP4CSUM_Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_
Tx,TSO6>
enabled=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_Tx,UDP4CSUM_
Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_Tx,TSO6>
address: 00:0c:bd:05:a2:fe
media: Ethernet autoselect (none)
status: no carrier
inet6 fe80::20c:bdff:fe05:a2fe%wm2 prefixlen 64 scopeid 0x5
wm3: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 9000
capabilities=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_
Tx,UDP4CSUM_Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_
Tx,TSO6>
enabled=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_Tx,UDP4CSUM_
Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_Tx,TSO6>
address: 00:0c:bd:05:a2:ff
media: Ethernet autoselect (none)
status: no carrier
inet6 fe80::20c:bdff:fe05:a2ff%wm3 prefixlen 64 scopeid 0x6
wm4: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 9000
capabilities=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_
Tx,UDP4CSUM_Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_
Tx,TSO6>
enabled=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_Tx,UDP4CSUM_
Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_Tx,TSO6>
address: 00:0c:bd:05:a3:00
media: Ethernet autoselect (none)
status: no carrier
inet6 fe80::20c:bdff:fe05:a300%wm4 prefixlen 64 scopeid 0x7
wm5: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 9000
capabilities=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_
Tx,UDP4CSUM_Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_
Tx,TSO6>
enabled=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_Tx,UDP4CSUM_
Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_Tx,TSO6>
address: 00:0c:bd:05:a3:01
```

```
media: Ethernet autoselect (none)
status: no carrier
inet6 fe80::20c:bdff:fe05:a301%wm5 prefixlen 64 scopeid 0x8
wm6: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 9000
capabilities=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_
Tx,UDP4CSUM_Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_
Tx,TSO6>
enabled=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_Tx,UDP4CSUM_
Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_Tx,TSO6>
address: 00:0c:bd:05:a3:02
media: Ethernet autoselect (none)
status: no carrier
inet6 fe80::20c:bdff:fe05:a302%wm6 prefixlen 64 scopeid 0x9
wm7: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 9000
capabilities=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_
Tx,UDP4CSUM_Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_
Tx,TSO6>
enabled=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_Tx,UDP4CSUM_
Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_Tx,TSO6>
address: 00:0c:bd:05:a3:03
media: Ethernet autoselect (none)
status: no carrier
inet6 fe80::20c:bdff:fe05:a303%wm7 prefixlen 64 scopeid 0xa
wm8: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 9000
capabilities=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_
Tx,UDP4CSUM_Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_
Tx,TSO6>
enabled=7ff80<TSO4,IP4CSUM_Rx,IP4CSUM_Tx,TCP4CSUM_Rx,TCP4CSUM_Tx,UDP4CSUM_
Rx,UDP4CSUM_Tx,TCP6CSUM_Rx,TCP6CSUM_Tx,UDP6CSUM_Rx,UDP6CSUM_Tx,TSO6>
address: 00:1e:67:6d:f6:ee
media: Ethernet autoselect (100baseTX full-duplex)
status: active
inet6 fe80::21e:67ff:fe6d:f6ee%wm8 prefixlen 64 scopeid 0xb
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33172
inet 127.0.0.1 netmask 0xff000000
inet6 ::1 prefixlen 128
inet6 fe80::1%lo0 prefixlen 64 scopeid 0xc
eqcli diags>
```

netstat

The netstat command displays the network status information. An example of a `netstat` output is as follows:

```
eqcli diags> netstat
Netstat output:
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        State
tcp        0,0           0,0       192.168.5.90.29897     192.168.0.17.80
     SYN_SENT
tcp        0,0           0,0       172.16.5.93.443        *.*
     LISTEN
tcp        0,0           0,0       172.16.5.94.443        *.*
     LISTEN
tcp        0,0           0,0       172.16.5.92.443        *.*
     LISTEN
tcp        0,0           0,0       172.16.5.90.22         *.*
     LISTEN
tcp        0,0           0,0       172.16.5.90.80         *.*
     LISTEN
tcp        0,0           0,0       127.0.0.1.92           *.*
     LISTEN
tcp        0,0           0,0       127.0.0.1.91           *.*
     LISTEN
tcp        0,0           0,0       127.0.0.1.90           *.*
     LISTEN
tcp        0,0           0,0       172.16.5.97.80         *.*
     LISTEN
tcp        0,0           0,0       172.16.5.91.80         *.*
     LISTEN
udp        0,0           0,0       *.*                    *.*
udp        0,0           0,0       *.*                    *.*
udp        0,0           0,0       *.*                    *.*
udp        0,0           0,0       172.16.5.90.65533      *.*
udp        0,0           0,0       172.16.5.90.161        *.*
Active Internet6 connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
udp6       0      0  *.*                    *.*
udp6       0      0  *.*                    *.*
udp6       0      0  *.*                    *.*
Active UNIX domain sockets
Address  Type    Recv-Q Send-Q    Inode     Conn     Refs  Nextref Addr
6ac42afc stream      0      0        0 6ac42b4c        0        0
/tmp/.AgentSockets/A
67b028cc stream      0      0        0 69b0fc44        0        0
/tmp/.AgentSockets/A
6ab04d24 stream      0      0 6ab16790        0        0        0
/tmp/.AgentSockets/A
6ac42b4c stream      0      0        0 6ac42afc        0        0 ->
/tmp/.AgentSockets/A
```

```
69b0fc44 stream      0      0        0 67b028cc        0        0 ->
/tmp/.Agent-- press space for mSockets/A
6ae0d7e4 dgram       0      0        0 66c3d5f4        0 6ab04234 ->
/var/run/log
69b0fc94 dgram       0      0 6b474b00        0        0        0
/tmp/eqsock/anon-cli_diag.5908-0
6b182e64 dgram       0      0        0 66c3d5f4        0 6ae0d294 ->
/var/run/log
6b182eb4 dgram       0      0 6b1949a0        0        0        0
/tmp/eqsock/anon-vlbd.1265-0
6ae0d294 dgram       0      0        0 66c3d5f4        0 6ae0d424 ->
/var/run/log
6ae0d424 dgram       0      0        0 66c3d5f4        0 6ae0d5b4 ->
/var/run/log
6ae0d474 dgram       0      0 6b14b638        0        0        0
/tmp/eqsock/anon-envoy_agent.97-0
6ae0d744 dgram       0      0        0 66c3d5f4        0 6ae0d924 ->
/var/run/log
6ae0d924 dgram       0      0        0 66c3d5f4        0 6ab04374 ->
/var/run/log
6ae0db04 dgram       0      0 6adfa584        0        0        0
/tmp/eqsock/anon-smartd.990-0
6ae0da64 dgram       0      0 6adfa214        0        0        0
/tmp/eqsock/anon-envoy.418-0
6ae0d834 dgram       0      0 6ae17798        0        0        0
/tmp/eqsock/anon-l3pd.1204-0
6ae0da14 dgram       0      0 6adfa0b4        0        0        0
/tmp/eqsock/anon-statsd.1091-0
6ab04234 dgram       0      0        0 66c3d5f4        0 6b182e64 ->
/var/run/log
6ac42a5c dgram       0      0 6ad3f168        0        0        0
/tmp/eqsock/snmptrap.sock
6ab043c4 dgram       0      0        0 66c3d5f4        0 6ab04eb4 ->
/var/run/log
6ac428cc dgram       0      0 67b2f164        0        0        0
/tmp/eqsock/anon-t_roxy_n3.1059-0
6ab048c4 dgram       0      0 6ab72dc4        0        0        0
/tmp/eqsock/anon-t_roxy_n3.1129-0
69b0f0b4 dgram       0      0 6ab72bb4        0        0        0
/tmp/eqsock/anon-acvd.1082-0
6ab04a04 dgram       0      0 6ab72d14        0        0        0
/tmp/eqsock/anon-udppd.1061-0
6ab04b44 dgram       0      0 67b2f214        0        0        0
/tmp/eqsock/anon-lbmd.1078-0
6ab04c34 dgram       0      0        0 66c3d5f4        0 6ab04e14 ->
/var/run/log
6ab04af4 dgram       0      0 6ab16160        0        0        0
/tmp/eqsock/an-- press space for mon-lbmd.270-0
6ab04be4 dgram       0      0        0 66c3d5f4        0 6ab04c84 ->
/var/run/log
```

```
6ab04b94 dgram       0      0 6ab162c0        0        0        0
/tmp/eqsock/anon-lbmd.1075-0
6ab04c84 dgram       0      0        0 66c3d5f4        0 6ab04c34 ->
/var/run/log
6ab04e14 dgram       0      0        0 66c3d5f4        0 69b0f6f4 ->
/var/run/log
69b0f7e4 dgram       0      0        0 66c3d5f4        0 69b0f744 ->
/var/run/log
69b0f6f4 dgram       0      0        0 66c3d5f4        0 69b0f7e4 ->
/var/run/log
69b0fe24 dgram       0      0 6a94b16c        0        0        0
/tmp/eqsock/anon-peerd.767-2
66c3d8c4 dgram       0      0 6a94b21c        0        0        0
/tmp/eqsock/anon-peerd.767-1
67b027dc dgram       0      0 6ab160b0        0        0        0
/tmp/eqsock/anon-acvd.1052-0
69b0f604 dgram       0      0 6a94b37c        0        0        0
/tmp/eqsock/anon-peerd.767-0
69b0f5b4 dgram       0      0        0 66c3d5f4        0 66c3d284 ->
/var/run/log
6ae0dd84 dgram       0      0 6adfa634        0        0        0
/tmp/eqsock/smartd.sock
67b0241c dgram       0      0        0 66c3d5f4        0 67b0273c ->
/var/run/log
6ab04f04 dgram       0      0 6ab16580        0        0        0
/tmp/eqsock/anon-cpsmib2agt.1021-0
6ac42aac dgram       0      0 6ad3f218        0        0        0
/tmp/eqsock/anon-cpsequalagt.207-0
69b0f744 dgram       0      0        0 66c3d5f4        0 69b0f5b4 ->
/var/run/log
69b0ff14 dgram       0      0 6aaeadd0        0        0        0
/tmp/eqsock/peerd.sock
69b0fe74 dgram       0      0 6aaeae80        0        0        0
/tmp/eqsock/anon-peerd.767-6
67b020fc dgram       0      0 6aaeaf30        0        0        0
/tmp/eqsock/anon-peerd.767-5
67b0205c dgram       0      0 6a94b00c        0        0        0
/tmp/eqsock/anon-peerd.767-4
69b0fec4 dgram       0      0 6a94b0bc        0        0        0
/tmp/eqsock/anon-peerd.767-3
67b024bc dgram       0      0 67b2f424        0        0        0
/tmp/eqsock/alertd.sock
67b0291c dgram       0      0 67b2f4d4        0        0        0
/tmp/eqsock/an-- press space for mon-alertd.484-2
67b0296c dgram       0      0 67b2f584        0        0        0
/tmp/eqsock/anon-alertd.484-1
67b029bc dgram       0      0 67b2f634        0        0        0
/tmp/eqsock/anon-alertd.484-0
67b02b4c dgram       0      0        0 66c3d5f4        0 67b02e6c ->
/var/run/log
```

```
67b0273c dgram       0      0        0 66c3d5f4        0 67b02b4c ->
/var/run/log
6ab04f54 dgram       0      0 6ab16210        0        0        0
/tmp/eqsock/anon-lbmd.1049-0
67b02e6c dgram       0      0        0 66c3d5f4        0 66c3d1e4 ->
/var/run/log
67b0269c dgram    1950      0 66e85f30        0        0        0
/tmp/eqsock/anon-switchd.488-0
6ab04eb4 dgram       0      0        0 66c3d5f4        0 6ab04be4 ->
/var/run/log
66c3d144 dgram       0      0 67af8160        0        0        0
/tmp/eqsock/configd.sock
6ae0d9c4 dgram       0      0 6ae17f28        0        0        0
/tmp/eqsock/anon-hcd.96-0
69b0fd34 dgram       0      0 66e8100c        0        0        0
/tmp/eqsock/switchd.sock
66c3d1e4 dgram       0      0        0 66c3d5f4        0        0 ->
/var/run/log
6ae0d5b4 dgram       0      0        0 66c3d5f4        0 6ae0d744 ->
/var/run/log
69b0f014 dgram       0      0 6ab72c64        0        0        0
/tmp/eqsock/anon-acvd.1077-0
69b0fdd4 dgram       0      0 6a94b63c        0        0        0
/tmp/eqsock/anon--eqcli.562-0
6b182374 dgram       0      0        0 66c3d5f4        0 6ae0d7e4 ->
/var/run/log
6ab04374 dgram       0      0        0 66c3d5f4        0 6ab043c4 ->
/var/run/log
66c3d284 dgram       0      0        0 66c3d5f4        0 67b0241c ->
/var/run/log
6b472fac dgram       0      0 6b46b0c0        0        0        0
/tmp/eqsock/anon--eqcli.4994-0
66c3d5f4 dgram       0      0 67a869a8        0 6b182374        0
/var/run/log
6ac427dc dgram       0      0 6ad3f0b8        0        0        0
/tmp/eqsock/anon-t_roxy_n3.301-0
eqcli diags>
```

ps

The ps command displays information about all of the processes. An example of a ps output is as follows:

```
eqcli diags> ps
Ps output:
PID TTY   STAT    TIME COMMAND
0 ?       OKl  0:51.94 [system]
1 ?       Is   0:00.00 init
96 ?      S    0:00.16 /usr/local/libexec/hcd -F
97 ?      S    0:00.18 envoy_agent: Envoy Agent
207 ?     I    0:00.01 /usr/local/libexec/cpsequalagt
219 ?     Ss   1:42.00 /usr/sbin/syslogd -s -f /etc/syslog.conf -f
/etc/eq/sys
236 ?     I    0:00.01 /usr/sbin/sshd -D -f /etc/ssh/sshd_config -o
ListenAddr
270 ?     S    0:00.25 lbmd: LB Mgmt Daemon - l7_spirent
301 ?     Sl   0:00.10 /usr/local/sbin/t_roxy_n3 -F -m 20000 -M -I https-
test1
418 ?     S    0:00.19 envoy: Envoy
427 ?     Ss   0:09.78 configd: Configuration Management Daemon
484 ?     S    0:01.12 alertd: Alert Daemon
487 ?     Is   0:00.00 /usr/sbin/powerd
488 ?     S    1:29.62 switchd: -Switch Management daemon
516 ?     I    0:00.02 run_sntp -1 pool.ntp.org
767 ?     S    0:26.29 peerd: Peer Daemon
778 ?     Ss   0:00.04 /usr/sbin/cron
792 ?     Is   0:00.00 /usr/sbin/inetd -l
966 ?     I    0:00.00 /usr/libexec/httpd -U _eqcli -s -b -f -n -i
172.16.5.90
990 ?     S    0:00.02 smartd: Smart Control Daemon
1021 ?    I    0:00.00 /usr/local/libexec/cpsmib2agt
1047 ?    I    0:00.01 /usr/local/libexec/snmpdm -d -snmp_bindaddr
172.16.5.90
1049 ?    S    0:00.49 lbmd: LB Mgmt Daemon - l4_spirent_1
1052 ?    S    0:02.38 /usr/local/libexec/acvd -F l4_spirent
1059 ?    Sl   0:00.09 /usr/local/sbin/t_roxy_n3 -F -m 20000 -M -I https-
test3
1061 ?    I    0:00.00 /usr/local/libexec/udppd -F l4_spirent_dns
1075 ?    S    0:00.48 lbmd: LB Mgmt Daemon - l4_spirent_dns
1077 ?    S    0:00.91 /usr/local/libexec/acvd -F l7_spirent
1078 ?    S    0:00.48 lbmd: LB Mgmt Daemon - l4_spirent
1082 ?    S    0:02.42 /usr/local/libexec/acvd -F l4_spirent_1
1091 ?    S    0:04.62 statsd: Equalizer Statistics Recorder
1129 ?    Sl   0:00.08 /usr/local/sbin/t_roxy_n3 -F -m 20000 -M -I https-
test2
1204 ?    Il   0:01.38 /usr/local/libexec/l3pd -F
1265 ?    S    0:00.03 /usr/local/libexec/vlbd -F
4994 tty00 SNs+ 0:00.01 -eqcli
5343 tty00 ON+  0:00.00 ps -a -x
```

```
5725 tty00 SN+  0:00.00 less -E -n -X -P -- press space for more, 'q' to
quit -
562 ttyE0 INs+ 0:00.01 -eqcli
eqcli diags>
```

top

The `top` command displays the top processes on the system. An exmple of a `top` output is as follows:

```
eqcli diags> top
Top output:
load averages:  0.02,  0.01,  0.00;               up 0+06:50:58
20:06:32
39 processes: 37 sleeping, 2 on CPU
CPU00 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
idle
CPU01 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
idle
CPU02 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
idle
CPU03 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
idle
CPU04 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
idle
CPU05 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
idle
CPU06 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
idle
CPU07 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
idle
CPU08 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
idle
CPU09 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
idle
CPU10 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
idle
CPU11 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
idle
CPU12 states:  0.3% user,  0.0% nice,  0.1% system,  0.0% interrupt, 99.6%
idle
CPU13 states:  0.2% user,  0.0% nice,  0.3% system,  0.0% interrupt, 99.5%
idle
CPU14 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
idle
CPU15 states:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100%
```

```
idle
Memory: 61M Act, 400K Wired, 16M Exec, 11M File, 2217M Free
Swap:
PID USERNAME PRI NICE   SIZE   RES STATE      TIME   WCPU    CPU COMMAND
219 root      85    0  5672K 1720K kqueue/1   1:42  0.00%  0.00% syslogd
488 _switchd  85    0  6388K 3524K kqueue/1   1:30  0.00%  0.00% switchd
0 root        0    0     0K  544M CPU/15     0:47  0.00%  0.00% [system]
767 _peerd    85    0  6540K 3972K kqueue/1   0:26  0.00%  0.00% peerd
427 root      85    0    15M   11M kqueue/1   0:09  0.00%  0.00% configd
1091 _statsd  85    0  4896K 3096K select/1   0:04  0.00%  0.00% statsd
1082 _acvd    85    0  6288K 2860K kqueue/5   0:02  0.00%  0.00% acvd
1052 _acvd    85    0  6288K 2860K kqueue/1   0:02  0.00%  0.00% acvd
484 _alertd   85    0  6432K 3124K kqueue/1   0:01  0.00%  0.00% alertd
1204 _l3pd    85    0    15M 2588K netio/14   0:01  0.00%  0.00% l3pd
5068 _eqcli   33   10  3288K 1328K CPU/12     0:00  0.00%  0.00% top
1265 _vlbd    85    0    34M 5684K nanosl/1   0:00  0.00%  0.00% vlbd
1129 _roxy    85    0    13M 3520K kqueue/9   0:00  0.00%  0.00% t_roxy_n3
301 _roxy     85    0    13M 3520K kqueue/1   0:00  0.00%  0.00% t_roxy_n3
1059 _roxy    85    0    12M 3504K kqueue/1   0:00  0.00%  0.00% t_roxy_n3
97 _envoy_a   85    0  6908K 3360K kqueue/1   0:00  0.00%  0.00% envoy_
agent
418 _envoy    85    0  6880K 3348K kqueue/1   0:00  0.00%  0.00% envoy
1077 _acvd    85    0  6288K 2844K kqueue/1   0:00  0.00%  0.00% acvd
207 _snmp     85    0  6088K 2820K select/1   0:00  0.00%  0.00%
cpsequalagt
1021 _snmp    85    0  6112K 2804K select/1   0:00  0.00%  0.00% cpsmib2agt
236 root      85    0  5224K 2720K select/1   0:00  0.00%  0.00% sshd
1075 _lbmd    85    0  4528K 2520K select/1   0:00  0.00%  0.00% lbmd
1049 _lbmd    85    0  4528K 2520K select/1   0:00  0.00%  0.00% lbmd
1078 _lbmd    85    0  4528K 2520K select/8   0:00  0.00%  0.00% lbmd
270 _lbmd     85    0  4512K 2480K select/9   0:00  0.00%  0.00% lbmd
1061 _udppd   85    0  4180K 2132K kqueue/1   0:00  0.00%  0.00% udppd
1047 _snmp    85    0  3016K 2076K select/1   0:00  0.00%  0.00% snmpdm
96 _hcd       85    0  4216K 2016K kqueue/1   0:00  0.00%  0.00% hcd
990 _switchd  85    0  4000K 1952K kqueue/1   0:00  0.00%  0.00% smartd
966 _eqcli    85    0  5272K 1764K select/1   0:00  0.00%  0.00% httpd
516 root      85    0  3056K 1440K wait/14    0:00  0.00%  0.00% run_sntp
5850 ntpd     85    0  4912K 1416K select/1   0:00  0.00%  0.00% sntp
778 root      85    0  3088K 1204K nanosl/1   0:00  0.00%  0.00% cron
1 root        85    0  3096K 1140K wait/13    0:00  0.00%  0.00% init
792 root      85    0  3140K  972K kqueue/1   0:00  0.00%  0.00% inetd
487 root      85    0  3120K  896K kqueue/1   0:00  0.00%  0.00% powerd
4994 eqadmin  80   10  6380K 3796K wait/12    0:00  0.00%  0.00% eqcli
562 eqadmin   80   10  6380K 2816K ttyraw/1   0:00  0.00%  0.00% eqcli
5738 eqadmin  80   10  3164K 1164K pipe_r/1   0:00  0.00%  0.00% less
eqcli diags>
```

Refer to "Using tcpdump" on page 870 for descriptions of this utility.

Using tcpdump

Note - You must have administrator privileges on your Equalizer to use the tcpdump feature.

tcpdump is a packet analyzer tool that can be used to analyze Equalizer packet activity to/from:

- an interface (port)
- an aggregated interface
- VLAN
- cluster
- server

It prints the contents of network packets and allows you to intercept and display TCP/IP and other packets being transmitted or received over the network on which the appliance is installed. It. prints out a description of the contents of packets that match Boolean expressions and saves the packet data to a `*.tgz` file stored in `.../var/crash` in the Equalizer file system. The file can then be used for later analysis. You can capture packets from a maximum of 5 objects at one time.

The flexibility provided by Equalizer's tcpdump feature is that you can capture packets based on header information, capture packets to/from servers or cluster and ports, or capture packets to/from objects based on protocol, such as ICMP.

In all cases, only packets that match expressions will be processed by tcpdump.

tcpdump is used with the Equalizer CLI using the `eqcli > diags tcpdump` commands or in `diags` context. The number of packets captured can be specified by either command line syntax or by manually halting a capture-in-progress using **CTRL+C** to stop it. For example, if you need to capture packets from a server (**`sv01`**) you would enter the following:

```
eqcli diags > tcpdump count 50 capture server sv01
```

In this example, tcpdump will capture 50 packets to/from server **`sv01`** and store the capture to `.../var/crash` in the Equalizer file system. Since the number of packets to capture is specified, it is not necessary to use **CTRL+C** to stop the capture.

Internally, Equalizer stores up to 10MB in up to 10-1MB raw packet capture files. That is, 1MB files are filled with capture data until a maximum of 10 files are full. When the 10 files are full, incoming captures will overwrite the first 1MB file, then the 2nd, and, so on. In the event that a packet count is not specified in the CLI syntax, this mechanism prevents captured data from exceeding Equalizer's 10MB capacity.

The tcpdump files that will be stored in the Equalizer file system will be in the following format:

tcpdump_objecttypeobjectname-tcp-pcap_MM_DD_YY_HH-MM{AM|PM}.tgz

where `objecttype` can be;

- `iface` - an interface
- `agr` - an aggregated interface
- `sv`- a server
- `vlan`- a vlan
- `cl`- a cluster

The time stamp in the file name is the time that the file was generated.

Foreground Feature

You also have the option of printing the output of a tcpdump capture to your screen, rather than to a pcap, Using the `fg` command within the CLI syntax, this option you can capture one instance at a time. In the example below, 10 packets are to be captured from a `cluster cl-http`.

```
eqcli > diags tcpdump fg count 10 capture cluster cl-http

tcpdump: Press Ctrl+C to quit.
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wm0, link-type EN10MB (Ethernet), capture size 65535 bytes
12:33:35.312242 ARP, Request who-has 172.16.166.10 tell 172.16.128.1,
length 46
12:33:35.312253 ARP, Reply 172.16.166.10 is-at 00:90:0b:29:89:88 (oui
Unknown),
length 28
12:33:35.312342 IP 192.168.10.19.49749 > 172.16.166.10.http: Flags [S], seq
1452094800, win 5840, options [mss 1460,sackOK,TS val 6931863 ecr
0,nop,wscale
6], length 0
12:33:35.312374 IP 172.16.166.10.http > 192.168.10.19.49749: Flags [S.],
seq
771217372, ack 1452094801, win 46417, options [mss 1460,nop,wscale
4,nop,nop,TS
val 1 ecr 6931863,sackOK,nop,nop], length 0
12:33:35.313494 IP 192.168.10.19.49750 > 172.16.166.10.http: Flags [S], seq
1451122556, win 5840, options [mss 1460,sackOK,TS val 6931863 ecr
0,nop,wscale
6], length 0
12:33:35.313513 IP 172.16.166.10.http > 192.168.10.19.49750: Flags [S.],
seq
778147759, ack 1451122557, win 57213, options [mss 1460,nop,wscale
4,nop,nop,TS
val 1 ecr 6931863,sackOK,nop,nop], length 0
12:33:35.314834 IP 192.168.10.19.49752 > 172.16.166.10.http: Flags [S], seq
```

```
1457919162, win 5840, options [mss 1460,sackOK,TS val 6931863 ecr
0,nop,wscale
6], length 0
12:33:35.314835 IP 192.168.10.19.49751 > 172.16.166.10.http: Flags [S], seq
1459230960, win 5840, options [mss 1460,sackOK,TS val 6931863 ecr
0,nop,wscale
6], length 0
12:33:35.314842 IP 192.168.10.19.49753 > 172.16.166.10.http: Flags [S], seq
1461671172, win 5840, options [mss 1460,sackOK,TS val 6931863 ecr
0,nop,wscale
6], length 0
12:33:35.314847 IP 172.16.166.10.http > 192.168.10.19.49751: Flags [S.],
seq
793126319, ack 1459230961, win 39153, options [mss 1460,nop,wscale
4,nop,nop,TS
val 1 ecr 6931863,sackOK,nop,nop], length 0
10 packets captured
409 packets received by filter
0 packets dropped by kernel
12000004: You have 8 pending alert notifications.
eqcli >
```

```
eqcli > diags tcpdump fg count 10 capture cluster cl-http

tcpdump: Press Ctrl+C to quit.
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wm0, link-type EN10MB (Ethernet), capture size 65535 bytes
12:33:35.312242 ARP, Request who-has 172.16.166.10 tell 172.16.128.1,
length 46
12:33:35.312253 ARP, Reply 172.16.166.10 is-at 00:90:0b:29:89:88 (oui
Unknown),
length 28
12:33:35.312342 IP 192.168.10.19.49749 > 172.16.166.10.http: Flags [S], seq
1452094800, win 5840, options [mss 1460,sackOK,TS val 6931863 ecr
0,nop,wscale
6], length 0
12:33:35.312374 IP 172.16.166.10.http > 192.168.10.19.49749: Flags [S.],
seq
771217372, ack 1452094801, win 46417, options [mss 1460,nop,wscale
4,nop,nop,TS
val 1 ecr 6931863,sackOK,nop,nop], length 0
12:33:35.313494 IP 192.168.10.19.49750 > 172.16.166.10.http: Flags [S], seq
1451122556, win 5840, options [mss 1460,sackOK,TS val 6931863 ecr
0,nop,wscale
6], length 0
12:33:35.313513 IP 172.16.166.10.http > 192.168.10.19.49750: Flags [S.],
seq
778147759, ack 1451122557, win 57213, options [mss 1460,nop,wscale
4,nop,nop,TS
val 1 ecr 6931863,sackOK,nop,nop], length 0
12:33:35.314834 IP 192.168.10.19.49752 > 172.16.166.10.http: Flags [S], seq
1457919162, win 5840, options [mss 1460,sackOK,TS val 6931863 ecr
0,nop,wscale
6], length 0
12:33:35.314835 IP 192.168.10.19.49751 > 172.16.166.10.http: Flags [S], seq
1459230960, win 5840, options [mss 1460,sackOK,TS val 6931863 ecr
0,nop,wscale
6], length 0
12:33:35.314842 IP 192.168.10.19.49753 > 172.16.166.10.http: Flags [S], seq
1461671172, win 5840, options [mss 1460,sackOK,TS val 6931863 ecr
0,nop,wscale
6], length 0
12:33:35.314847 IP 172.16.166.10.http > 192.168.10.19.49751: Flags [S.],
seq
793126319, ack 1459230961, win 39153, options [mss 1460,nop,wscale
4,nop,nop,TS
val 1 ecr 6931863,sackOK,nop,nop], length 0
10 packets captured
409 packets received by filter
0 packets dropped by kernel
12000004: You have 8 pending alert notifications.
eqcli >
```

Using Custom Filtering Expressions

Custom filtering expressions can be used in the tcpdump CLI syntax that allow you to trim out various types of traffic. You can combine them in different ways to find exactly what you're looking for. PCAP filtering expressions are used in these cases. Refer to www.tcpdump.org for detailed descriptions of using PCAP filtering expressions.

When using custom PCAP filtering expressions, the `expr` command is used in the CLI syntax, For example,

```
eqcli diags> tcpdump capture iface|agr|vlan|cluster|server expr pcap filter
expression
```

Examples are provided below that show header-based filtering, host-based filtering, port filtering, and protocol filtering. As described above, the resultant `tcpdump_objecttypeobjectname-tcp-pcap_MM_DD_YY_HH-MM{AM|PM}.tgz` file will be stored in Equalizer's file system in `.../var/crash`.

Examples

Expressions select which packets will be included in the packet capture. If no expression is given, ALL packets on the network with are from/to the specified VLAN/interface(port)/aggregated interface/cluster/server will be included. Otherwise, only packets for which the expression is "true" will be captured.

Header based filtering;

1. Capture only the SYN packets on an interface **`if01`**:

```
eqcli-diags> tcpdump capture iface if01 expr “tcp[13] &2!=0”
```

2. Capture only SYN or FIN packets on a vlan **`vl01`**:

```
eqcli-diags> tcpdump capture vlan vl01 expr “tcp[13] &3!=0”
```

3. Capture all packets to/from a cluster **`cl01`** that are larger than 576 bytes:

```
eqcli-diags> tcpdump capture cluster cl01 expr “ip[2:2] > 576”
```

Host based filtering;

1. Capture traffic between a server **`sv01`** and a host with IP **`#.#.#.#`**:

```
eqcli-diags> tcpdump capture server sv01 expr “host #.#.#.#”
```

Filtering Ports;

1. Capture all packets to/from a cluster **`cl01`** and a port **`XX`**:

```
eqcli-diags> tcpdump capture cluster cl01 expr “port XX”
```

Protocol Filtering;

1. Capture all **`icmp`** packets to/from an interface instance **`if01`**:

```
eqcli-diags> tcpdump capture iface if01 expr “icmp”
```

Using Watchdog Timers

Watchdog timers are used to reboot the system if something goes wrong. When working properly, a failover pair will continue functioning because the backup system can take over when the primary system is rebooted.

Types of Watchdogs

CPU reset	This is essentially a "power cycling" of the CPU. It's a hardware operation, so the state of the software doesn't matter. It will always succeed.
reboot	As opposed to a CPU-reset, a reboot may not succeed - if the operating system hangs, for example.
panic	A reboot that is caused by a kernel exception, or by Non-maskable Interrupt (NMI). This default behavior can also be changed to cause the system to enter the kernel debugger instead of rebooting.
Non-maskable Interrupt (NMI)	In this scenario, the operating system stops performing when it receives such a signal. For Equalizer systems, an NMI causes a panic.

Watchdog Timers

Timer Type	**Description**
ichlpcib0	This is a PCI hardware watchdog timer. It is generally immediately available after boot, however, may be unusable because of hardware limitations. In such an instance, the kernel prints out a message to the boot log which loads the software watchdog instead. This watchdog has a single timer, and causes a CPU reset when the watchdog has not been tickled for <timer> seconds.

Timer Type	Description
ipmi0	This is an Intelligent Platform Management Interface (IPM)I hardware watchdog timer. It is not available until several seconds after the boot process is complete. There are two timers for this watchdog: an *NMI timer* and a *reset timer*. An *NMI timer* is generated when the watchdog has not been tickled for <NMI timer> seconds. A CPU *reset* is generated when the watchdog has not been tickled for <reset timer> seconds. The idea is that you can configure your system to panic (using an NMI watchdog) and then several seconds later reboot (using a CPU reset). The following rules apply: reset = 0, nmi => 0: the watchdog is not armed (i.e. if the reset timer is 0, the nmi value does not matter) reset > 0, nmi = 0: The system will reset the CPU <reset> seconds after the timer stops being tickled. (This is the default behavior, reset = 30). reset > 0, nmi > 0, reset < nmi: The system will generate an NMI <nmi> seconds after the timer stops being tickled. No CPU reset is ever asserted. reset > 0, nmi > 0, reset > nmi: The system will generate an NMI <nmi> seconds after the timer stops being tickled. It will reset the CPU <reset> seconds after the timer stops being tickled. (Note: Not <reset> seconds after <nmi>). The NMI timer is controlled via the sysctl: hw.ipmi0.pretimeout. swwdog0: This is a NetBSD software watchdog. It is always available, however, requires the kernel to be operational. If the kernel is "locked up", the watchdog may not fire. It has a single timer, and causes a panic when it has not been tickled for <timer> seconds.
ichlpcib0	This is a PCI hardware watchdog timer. It is generally immediately available after boot, however, may be unusable because of hardware limitations. In such an instance, the kernel prints out a message to the boot log which loads the software watchdog instead. This watchdog has a single timer, and causes a CPU reset when the watchdog has not been tickled for <timer> seconds.

Timer Type	**Description**
ipmi0	This is an Intelligent Platform Management Interface (IPM)I hardware watchdog timer. It is not available until several seconds after the boot process is complete. There are two timers for this watchdog: an *NMI timer* and a *reset timer*. An *NMI timer* is generated when the watchdog has not been tickled for <NMI timer> seconds. A CPU *reset* is generated when the watchdog has not been tickled for <reset timer> seconds. The idea is that you can configure your system to panic (using an NMI watchdog) and then several seconds later reboot (using a CPU reset). The following rules apply: reset = 0, nmi => 0: the watchdog is not armed (i.e. if the reset timer is 0, the nmi value does not matter) reset > 0, nmi = 0: The system will reset the CPU <reset> seconds after the timer stops being tickled. (This is the default behavior, reset = 30). reset > 0, nmi > 0, reset < nmi: The system will generate an NMI <nmi> seconds after the timer stops being tickled. No CPU reset is ever asserted. reset > 0, nmi > 0, reset > nmi: The system will generate an NMI <nmi> seconds after the timer stops being tickled. It will reset the CPU <reset> seconds after the timer stops being tickled. (Note: Not <reset> seconds after <nmi>). The NMI timer is controlled via the sysctl: hw.ipmi0.pretimeout. swwdog0: This is a NetBSD software watchdog. It is always available, however, requires the kernel to be operational. If the kernel is "locked up", the watchdog may not fire. It has a single timer, and causes a panic when it has not been tickled for <timer> seconds.

User Options

There are three hidden eqcli options to control the behavior:

1. **`debug >configd ipmi_nmi <seconds>`**

   This sets the IPMI NMI timer. It has no effect on systems that do not have an IPMI (`ipmi0`) watchdog timer. The default value is 0.

2. **`debug > configd require_hw_wd <0 or 1>`**

   Defers loading the network subsystem until after a hardware watchdog has been armed. Has no effect if the watchdog has been completely disabled (reset timer is 0). Does not do anything until after the system is rebooted. The user will see "*Waiting for system processes...*" along with this message in the eq log: `20140127T190024|configd|w|04007413`:

   Deferring the system startup until the hardware watchdog timer becomes available applies to both the PCI (`ichlpcib0`)and IPMI (`ipmi0`) watchdogs timers, although the PCI watchdog should always be loaded on the first try if it is available and usable in the system. The default value is 0.

3. **`debug > configd debugger_on_panic <0 or 1>`**

   Sets the `ddb.onpanic sysctl`. The default is 0. If set to 1, the system enters the debugger on panic or on NMI. Therefore, if set and the IPMI watchdog NMI timer is active, the system will enter the debugger when the watchdog expires instead of a panic.

Note - The "reset" timer is controlled by the already-existing "hidden watchdog <seconds>" command.

When the system boots, a message (or two messages) will be output to the O/S log:

```
Jan 27 19:00:08 FADC600E-PROTO root: Starting watchdog timer swwdog0 with interval 30.
...
Jan 27 19:00:47 FADC600E-PROTO root: Starting watchdog timer ipmi0 with interval 30.
```

All of these options are stored in the configuration file and are synched between failover peers.

An typical example is as follows:

The default behavior is:

NMI timer = 0
debugger_on_panic = 0
require_hw_wd = 0
reset timer = 30 This means that if Equalizer "locks up", the IPMI watchdog will CPU-reset after 30 seconds. No debugging information will be available.

A customer that experiences such a problem may set up their system as follows:

NMI timer = 30
debugger_on_panic = 1
reset_timer = 29

The system will then drop to the debugger after 30 seconds, and remain there until physically rebooted/power cycled.

Note - The reset_timer can not be 0 in this configuration or the watchdog timer will not be armed! However, setting it to less than the NMI timer will keep the system from doing a CPU reset

If Equalizer has lockup issues during boot, it is possible that the system will begin processing traffic and then lock up. If the software watchdog is active at this time, it will never reboot -- which means that the standby failover peer will never take over as the primary unit. Setting the `require_hw_wd` option on these systems will prevent the system from processing traffic until after the IPMI watchdog is available. This will mean that if it locks up while the software watchdog is in use, it isn't processing traffic. If it locks up after it begins processing traffic, the IPMI watchdog will be in use so a CPU reset will work.

Configuring the Baseboard Management Controller (BMC)

Note - Currently, the E970LX, E670LX, and E470LX ADCs feature this optional utility.

The Equalizer E470LX, E670LX, and E970LX are equipped with a Baseboard Management Controller (BMC). This specialized service processor monitors the physical state of the ADC using sensors and communicates information to system administrators using an independent network connection. BMC is a specialized micro controller embedded on the motherboard of the Equalizer that manages the interface between the operating system and the hardware.

The BMC is active as long as your system is receiving power from an outlet and will remain ON, even if the system is shut down. It has a dedicated network interface and is accessible via the configured IP to perform power control functions using the Integrated BMC Web Console once it is configured.

Prerequisites

Before configuring the BMC utility it is recommended that you contact your network administrator to obtain:

- an IP address and subnet mask for the BMC
- the IP address of the gateway through which the BMC will be accessed

Configuration

The BMC is configured using CLI commands only. In the GUI, you can run the BMC commands shown in this section using the CLI widget on the Dashboard. Once initial configuration is complete, the BMC can be further configured through the Web Console.

This setup procedure describes the process of setting the initial user id and password and assigning an IP address on which to access the BMC web interface.

When you power up your system for the first time, you will not have connectivity with the BMC. You will need to:

1. Configure a password to use when logging in to the BMC console.
2. Enable the BMC to perform power control functions.
3. Configure an IP address (and Gateway) on which clients can communicate with it on the integrated BMC Web Console.

Proceed with the following after powering up your Equalizer:

1. Enter the following to view the initial BMC configuration:

```
eqcli > show bmc
User Name:      root
Status:         disabled
IP Address:     0.0.0.0/32 (static)
Gateway:        0.0.0.0

eqcli: 12000287: Operation successful
eqcli >
```

   The default **`User Name`** will be **`root`** when you use the CLI. You will be able to change this once you have set up network connectivity and have access to the Integrated BMC Web Console; however, only the user **`root`** can be configured using the CLI.

2. Configure a password to use with the `User Name` as follows. After entering **`bmc passwd`** at the eqcli prompt, press **ENTER** and follow the prompts to **`enter password`** and **`confirm password`**. press **ENTER** after entering and confirming.

```
eqcli > bmc passwd
Please enter password: password
Please confirm password: password

eqcli: 12000287: Operation successful
eqcli >
```

3. Enable the BMC by entering:

```
eqcli > bmc enable

eqcli: 12000287: Operation successful
eqcli >
```

4. Enter an IP Address (`lan ip`) for the BMC. The default IP address is `0.0.0.0/32`. This is a static address. You will also need a Gateway (`gw`) address if used; however, you can enter **`0.0.0.0`** as the gateway address if there is none.

```
eqcli > bmc lan ip 192.168.1.8/21 gw 192.168.0.1

eqcli: 12000287: Operation successful
eqcli >
```

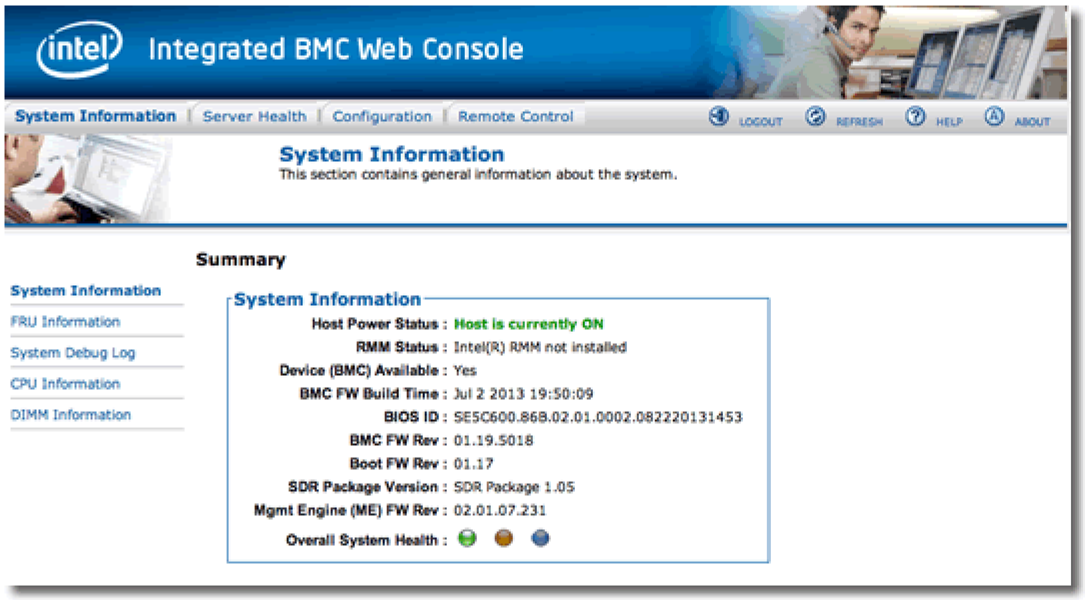
5. Confirm your configuration by entering the following:

```
eqcli > show bmc
User Name:      root
Status:         enabled
IP Address:     192.168.1.8/21 (static)
Gateway:        192.168.0.1

eqcli: 12000287: Operation successful
eqcli >
```

You should now be able to access the Integrated BMC Web Console for additional configuration options and for remote power control options. To access the console enter the IP address that you configured in the address bar of your web browser.

Log In to the Integrated BMC Web Console

To log in to the BMC Web Interface, enter the IP address into your browser's address box. The IP address is the one that you configured as the `lan ip` above. You will be prompted for the **User Name** and **Password** that you configured. When you click on the **Login** button, the **Integrated BMC Web Console** will be displayed as shown below:



System Information
This section contains general information about the system.

Summary

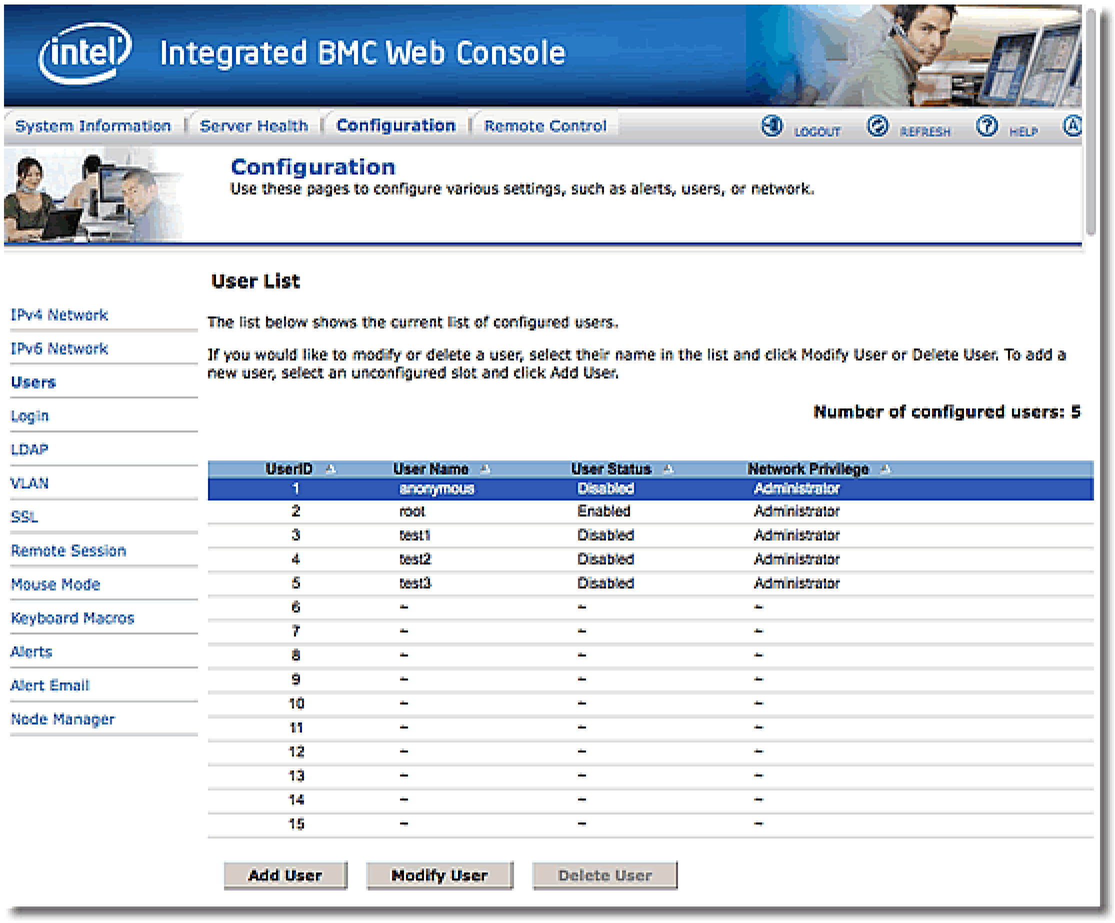
System Information

Host Power Status : Host is currently ON
RMM Status : Intel(R) RMM not installed
Device (BMC) Available : Yes
BMC FW Build Time : Jul 2 2013 19:50:09
BIOS ID : SE5C600.86B.02.01.0002.082220131453
BMC FW Rev : 01.19.5018
Boot FW Rev : 01.17
SDR Package Version : SDR Package 1.05
Mgmt Engine (ME) FW Rev : 02.01.07.231
Overall System Health : ● ● ●

Modifying the User Name, Password, User Status and Network Privileges

In order to use the Web Console as described in this section, you must have already set the BMC Console up for web acess by following the instructions in the previous section.

Once you have access to the Web Console and logged in, select the **Configuration** tab and then **Users** from the left navigational pane. The **User List** will appear on the right.



Configuration

Use these pages to configure various settings, such as alerts, users, or network.

User List

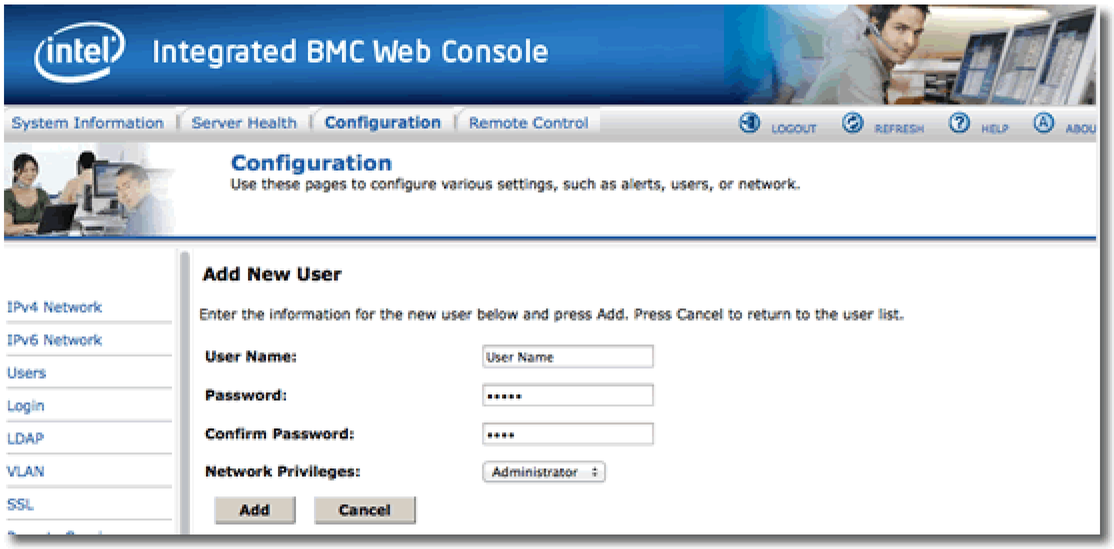
The list below shows the current list of configured users.

If you would like to modify or delete a user, select their name in the list and click Modify User or Delete User. To add a new user, select an unconfigured slot and click Add User.

Number of configured users: 5

UserID	User Name	User Status	Network Privilege
1	anonymous	Disabled	Administrator
2	root	Enabled	Administrator
3	test1	Disabled	Administrator
4	test2	Disabled	Administrator
5	test3	Disabled	Administrator
6	-	-	-
7	-	-	-
8	-	-	-
9	-	-	-
10	-	-	-
11	-	-	-
12	-	-	-
13	-	-	-
14	-	-	-
15	-	-	-

To add a **User**, click on the **Add User** button to display the **Add New User** screen shown below. Enter a **User Name** and **Password** and select the appropriate **Network Privileges** for the new user. Click the **Add** button when finished.



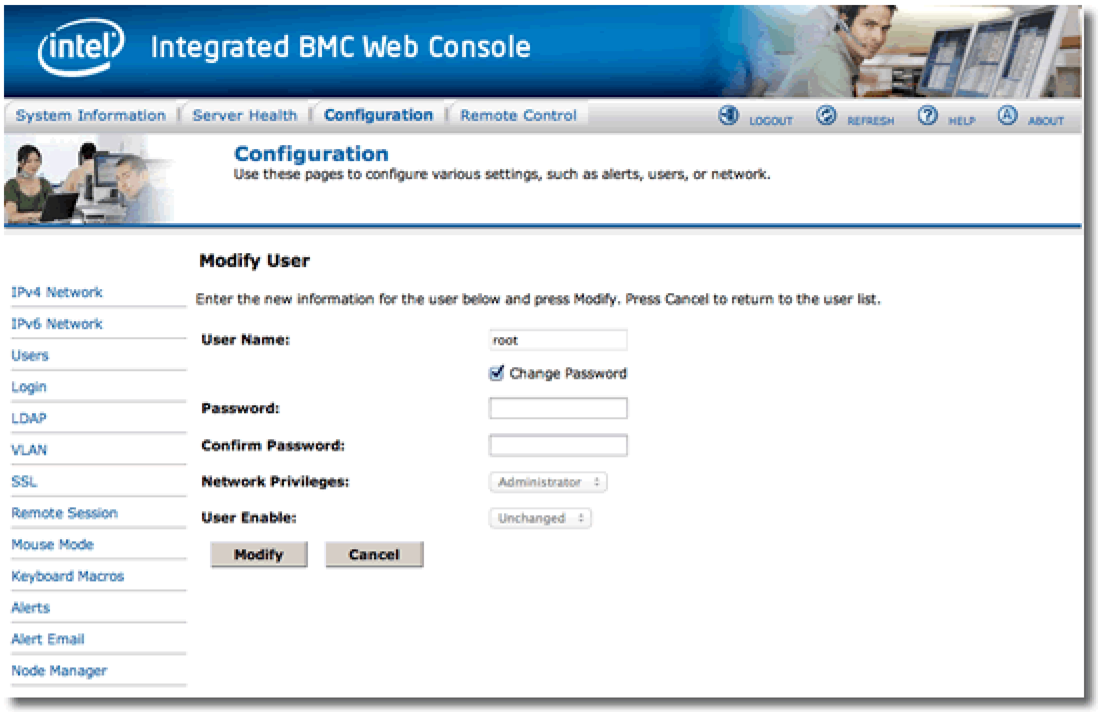
Configuration
Use these pages to configure various settings, such as alerts, users, or network.

Add New User

Enter the information for the new user below and press Add. Press Cancel to return to the user list.

User Name:	User Name
Password:	•••••
Confirm Password:	••••
Network Privileges:	Administrator

To modify an existing user's **User Name**, **User Status** or **Network Privileges**, select the user from the **User List** and click on the **Modify User** button. Click on the **Help** link at top right for instructions on how to use the Web Console interface.



Configuration
Use these pages to configure various settings, such as alerts, users, or network.

Modify User

Enter the new information for the user below and press Modify. Press Cancel to return to the user list.

User Name:	root
	✓ Change Password
Password:	
Confirm Password:	
Network Privileges:	Administrator
User Enable:	Unchanged

To change a user password in the web console, select the **Change Password** check box. Enter the Password and Confirm it and then click on the **Modify User** button.

Resetting a Forgotten BMC User Name and/or Password

If you forgot your password you will need to follow the procedure for configuring a *new* password using the CLI only. Log in to the CLI and configure a new password as follows:

1. Enter **`bmc passwd`** at the eqcli prompt and press **ENTER** .
2. Follow the prompts to **`enter password`** and **`confirm password`**; press **ENTER** after entering and confirming as shown below.

```
eqcli > bmc passwd
Please enter password: password
Please confirm password: password

eqcli: 12000287: Operation successful
eqcli >
```

Changing the BMC IP Address

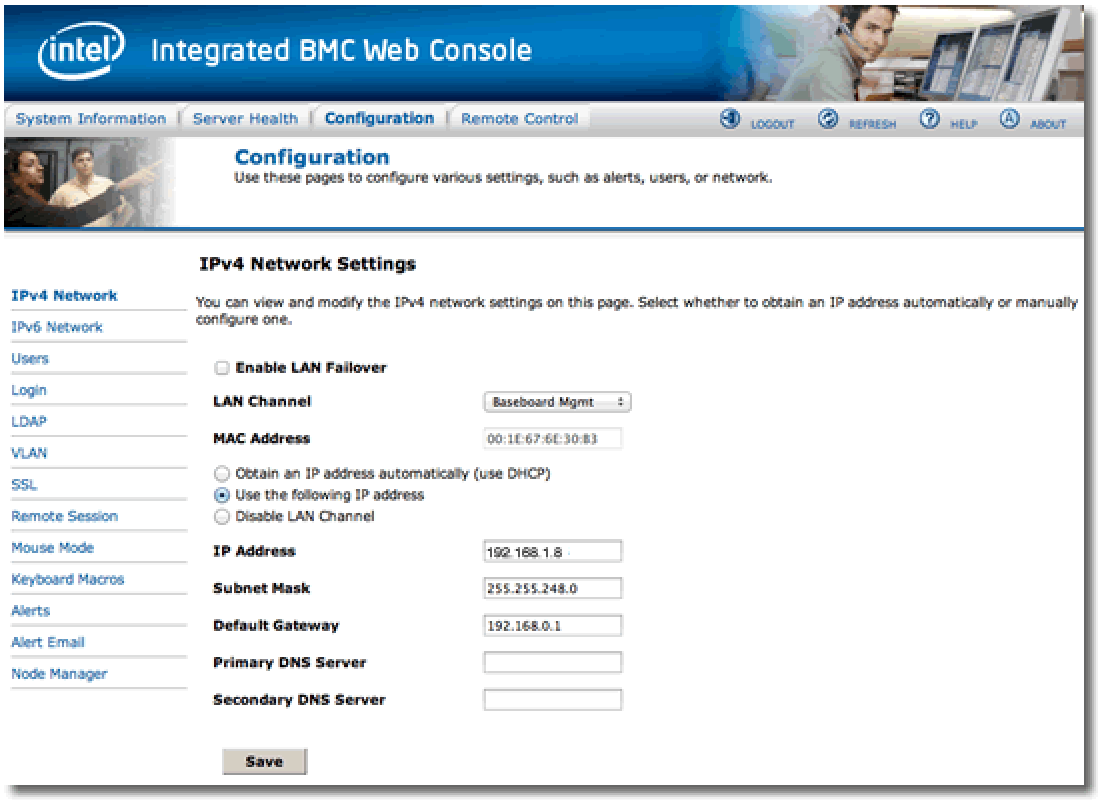
Note - Consult with your network administrator prior to making network changes.

The Web Console also provides you with the ability to change the BMC IP address. Click on the **Configuration** tab at the top and then click on **IPv4 Network** on the left navigational pane to display the following.

Make the necessary **IP Address**, **Subnet Mask**, and **Default Gateway** changes using values obtained from your network administrator. When you have finished, click on the **Save** button.

The BMC software does not check that a valid Subnet Mask value has been entered. Ensure that you have entered the correct Subnet Mask when configuring the BMC interface.

It is recommended that you leave the LAN Channel drop down selection as Baseboard Mgmt. as this is the selection that is configurable at this time.



Configuration

Use these pages to configure various settings, such as alerts, users, or network.

IPv4 Network Settings

You can view and modify the IPv4 network settings on this page. Select whether to obtain an IP address automatically or manually configure one.

☐ Enable LAN Failover

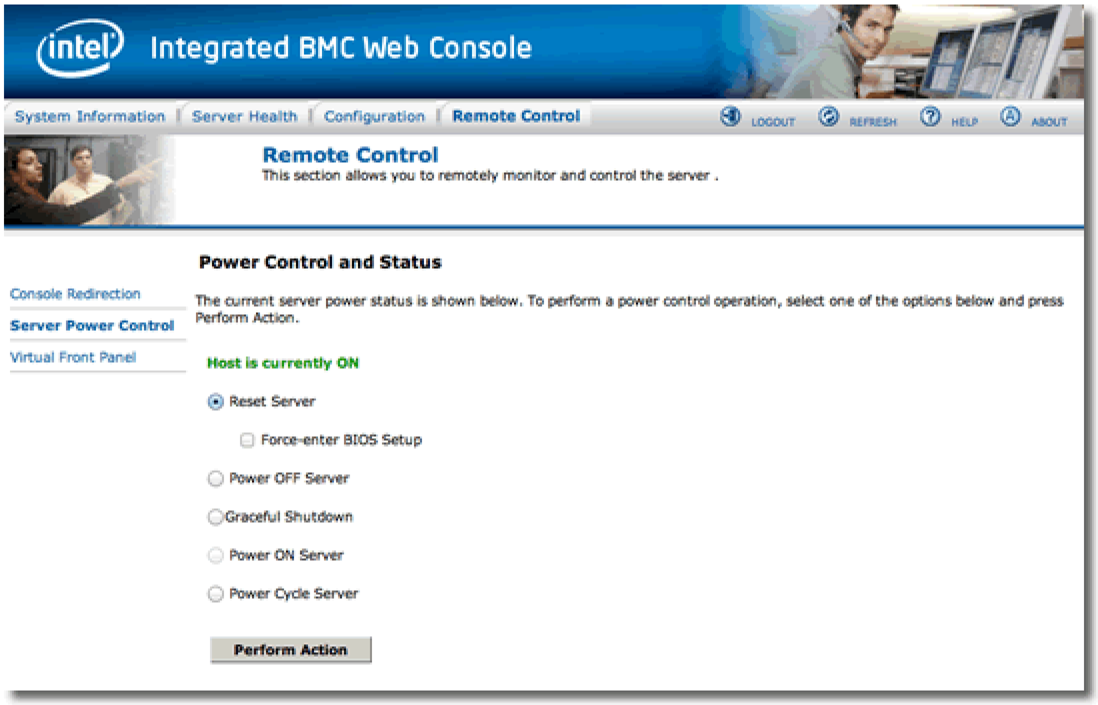
LAN Channel	Baseboard Mgmt
MAC Address	00:1E:67:6E:30:83

○ Obtain an IP address automatically (use DHCP)
[✓ Use the following IP address]
○ Disable LAN Channel

IP Address	192.168.1.8
Subnet Mask	255.255.248.0
Default Gateway	192.168.0.1
Primary DNS Server	
Secondary DNS Server	

Server Power Control

To access the power control interface, select the **Remote Control** tab at the top of the screen and then select **Server Power Control** on the left navigational pane. This will display the **Power Control and Status** screen shown below.



Remote Control

This section allows you to remotely monitor and control the server .

Power Control and Status

The current server power status is shown below. To perform a power control operation, select one of the options below and press Perform Action.

Host is currently ON

[✓ Reset Server]

Force-enter BIOS Setup

Power OFF Server

Graceful Shutdown

Power ON Server

Power Cycle Server

The following power control operations can be performed:

- **Reset Server** - Selecting this option will hard reset the host without powering off.
- **Force-enter BIOS Setup** - This option does not function without the use of the password required to enter the BIOS setup.
- **Power Off Server** - Selecting this option will immediately power off the host.
- **Graceful Shutdown** - Selecting this option will soft power off the host.
- **Power On Server** - Selecting this option will power on the host.
- **Power Cycle Server** - Selecting this option will immediately power off the host, then power it back on after one second.

All power control actions are done through the BMC and are immediate actions after clicking on the **Perform Action** button. It is suggested to gracefully shut down the operating system before initiating power actions.

Getting Additional Help on the BMC User Interface

Refer to the Intel documentation for additional information on using the BMC web interface.

Using IPMI to Power Servers On/Off

Note - Currently the E970LX, E670LX, and E470LX ADCs feature this utility.

The Intelligent Platform Management Interface (IPMI) is an open standard for software-based control of hardware functions, such as powering the system on and off. IPMI is implemented by a set of software tools (IPMItools) that communicate with the local machine or over a LAN connection.

Using IPMI, it is possible to power systems on and off. Using the Smart Control feature, you can configure your Equalizer to power "on" or "off", shut down, or reset at scheduled times, regular intervals, or when threshold values are reached. Additional information on Smart Control can be found in "Smart Control" on page 779.

In order to use an IPMI function to control a server, the server must have a Baseboard Management Controller (BMC), a separate network interface that provides IPMI services. The BMC is usually enabled and configured via the system BIOS, which must be accessed when the system boots. If the installed operating system on the server has an IPMI driver installed and configured, you may also be able to configure the BMC from the command line or using graphical utilities. The tools used to configure BMC controllers and IPMI drivers are specific to a server's hardware and OS platform. Refer to the hardware and operating system documentation for your servers for specific BMC and IPMI configuration instructions.

The BMC needs to be configured with the following information:

- The IP address on which to listen for IPMI requests.
- A user name and password.

The IP address, user name, and password specified when configuring the IPMI driver on a server must be provided as arguments to the IPMI functions used on the command line or with a Smart Control, so that Equalizer can log into the IPMI subsystem on the server.

IPMI commands are entered on the command line of the CLI from your Equalizer and entered using diagnostic commands. (Refer to "Diagnostic Commands" on page 179 for additional information on the use of these commands). They can also be used on the GUI in conjunction with Smart Control configuration.

Entering IPMI "Power" Commands Using the CLI

Enter IPMI commands using the following format:

```
eqcli >diags ipmi ip IP address user user passwd password cmd command
```

where:

`IP address` - is the IP address of the target server or BMC being issued the command.

`user` - is an enabled user name that is configured in the BMC or the target server or ADC.

`password` - is an enabled password that is configured in the BMC of the target server or ADC.

`command` - is the power command being used as shown below:

Power Command (cmd)	Operation
poweron	Entering this option will power on the server.
poweroff	Entering this option will immediately power off the server.
shutdown	Entering this option will soft power off the server.
reset	Entering this option will hard reset the server without powering off.

Configuring IPMI Power Controls using a Smart Control

IPMI support in Smart Controls has the following limitations:

- **The Smart Control interface does not allow the user to specify IPMI v1.5. Only IPMI v2.0 is supported at this time.**
- **The Smart Control interface does not allow the user to specify a key generator key. It always uses a "NULL" key generator key.**
- **The Smart Control interface does not allow the user to specify a cipher suite. It always uses cipher suite 3, which specifies RAKP-HMAC-SHA1 authentication, HMAC-SHA1-96 integrity, and AES-CBC-128 encryption algorithms.**

Appendix A
Equalizer OnDemand

Sections in this chapter include:

What is Equalizer OnDemand?

Equalizer OnDemand™ (EQOD) is a software-based virtual appliance that operates as an integral part of the virtual infrastructure model. EQOD is deployed as a single virtual server instance dedicated to load balancing and managing the application delivery needs of your business. The EQ/OS 10 platform on which Equalizer is built drives the robust application traffic management capabilities of the Virtual Equalizer. Envoy OnDemand SAE is a software based virtual appliance that provides you with the ability to send website traffic to the 'best' geographic location amongst several independent data centers, based on application health, static weighting, server load or lowest latency to the client, to ensure the best application response.

Equalizer instances can be deployed on a single server to maximize utilization of hardware infrastructure, and current server platforms can be used for load balancing and other application delivery needs without creating dependence on specific server hardware.

Equalizer OnDemand offers:

- Intelligent, layer 4-7 application-based load balancing
- Flexible hybrid network support
- Optimization for VMware virtualized environments; real-time resource metrics are obtained from VMware
- Scalability and availability for today’s complex mission critical virtualized applications
- Predictable, reliable, and consistent application performance
- All the cost and energy saving benefits of virtualization technology

EQOD can also be combined with Equalizer hardware appliances to best meet your specific application delivery challenges.

Differences from Equalizer Hardware

All load balancing functionality found in EQ/OS 10 running on an a virtual appliance is fully functional in EQOD. Some adjustments to functionality were necessary, however, in order to accommodate the VMware virtual machine environment.

1. EQOD is delivered with two Ethernet network interfaces configured -- one interface can be configured into a VLAN using port number **1**, the other using port number **2**. Port number 1 corresponds to the first interface added via VMware; port 2 to the second. The current release supports up to 16 network interfaces.
2. An interface port in the CLI or GUI can be assigned to one VLAN only, either:
   - a single untagged VLAN and , or
   - multiple tagged VLANs

   A port cannot be assigned to multiple untagged VLANs or to a mix of tagged and untagged VLANs.
3. EQOD is delivered with no serial console configured because this requires additional configuration by the user. A serial console can be added by editing the Virtual Machine settings. See your VMware product documentation for more information.

Adding Ports on VM Workstation

When adding an interface using VM Workstation (a.k.a. VM Player), you are not given the option to choose the type of network adapter added. as a result, you must edit the "vmx" file for the Virtual Machine manually and restart the Virtual Machine to see the new interface in the Equalizer OnDemand CLI and GUI.

1. Add an interface to the Equalizer Virtual Machine using the standard VM Workstation GUI controls.
2. Power off the Virtual Machine.
3. Find and edit the "*.vmx*" file for the Virtual Machine. Assuming that you have an image shipped with 2 interfaces and you are adding a third, you will find lines like this towards the end of the "*.vmx*" file:
4. Each interface has a set of properties that begin with "`ethernetN.`" where '*N*' is the interface number (this number indicates the order in which the interfaces were added to the VM). The first two interfaces have a line (highlighted in green) that indicates the network interface device type. The text highlighted in yellow is what VMware added to the file for the third interface. Note that there is no `virtualDev` line in this set of properties that indicates the interface type. This line needs to be added for the interface to work on Equalizer. The text below shows what the "`ethernet2.`" set of properties should look like after editing:

```
ethernet0.present = "TRUE"
ethernet0.virtualDev="e1000"
ethernet0.connectionType = "bridged"
ethernet0.wakeOnPcktRcv = "FALSE"
ethernet0.addressType = "generated"
ethernet0.linkStatePropagation.enable = "TRUE"
ethernet0.generatedAddress = "00:0c:29:bc:74:82"
ethernet0.pciSlotNumber = "32"
ethernet0.generatedAddressOffset = "0"
ethernet0.vnet = "vmnet2"
ethernet0.bsdName = "en0"
ethernet0.displayName = "Ethernet"
ethernet1.present = "TRUE"
ethernet1.virtualDev="e1000"
ethernet1.connectionType = "bridged"
ethernet1.wakeOnPcktRcv = "FALSE"
ethernet1.addressType = "generated"
ethernet1.linkStatePropagation.enable = "TRUE"
ethernet1.generatedAddress = "00:0c:29:bc:74:8c"
ethernet1.pciSlotNumber = "35"
ethernet1.generatedAddressOffset = "10"
ethernet1.vnet = "vmnet3"
ethernet1.bsdName = "en1"
ethernet1.displayName = "Ethernet2"
ethernet2.present = "TRUE"
ethernet2.wakeOnPcktRcv = "FALSE"
ethernet2.addressType = "generated"
ethernet2.generatedAddress = "00:0c:29:bc:74:96"
ethernet2.pciSlotNumber = "37"
ethernet2.generatedAddressOffset = "20"
```

5. Save your edits to the file.
6. Start the Equalizer VM and log in to the CLI or GUI. You should now see 3 interfaces ports when you run the `show interface` command in the CLI and when you open the **Interfaces** tab in the GUI.

Installing and Upgrading Equalizer OnDemand

VMware Host Requirements

The EQOD runs under any VMWare Hypervisors which support Version 8 virtual machines, including the following VMware releases:

- VMware ESX and ESXi 5.0 and higher
- VMware Fusion 4.X and higher
- VMware Workstation 8.X and higher
- VMware Player 4.X and higher

A VM instance of Equalizer requires the following minimum hardware resources:

- 1GB RAM
- 1GB free disk space
- 1 VMware supported 10/100/1000Gb Network Adapter (e1000 adapter type)
- Internet connectivity for license validation

The above requirements are *in addition to* the resources required to run VMware. See the VMware documentation for the VMware product you are using for installation requirements.

Installing EQOD Using OVF

VMware vSphere or vCenter Clients

When installing a new instance of Equalizer OnDemand onto an ESX or ESXi VMware server, you must begin the installation using the supplied OVF file as instructed in this guide. Installing Equalizer OnDemand by deploying the VMDK or VMX file directly without using the OVF file will lead to networking issues after the install is complete and is not a supported deployment method

VMware ESX and ESXi servers are managed using either the vSphere or vCenter management clients. If you are using either of these products, the OVF can be installed directly from the FTP site following the directions below.

1. Open the vSphere or vCenter client.
2. If your client has Internet access, do the following; otherwise goto the next step.
   a. Select **File > Deploy OVF Template**.
   b. Click **Deploy from URL**, and enter the FTP URL for the OVF image: Click **Next.** You can copy the hyperlink target available on the EQOD section of the support page at:

      http://www.coyotepoint.com/content/eqos-10-support-page
   c. The OVF details are displayed. Click **Next.**
   d. Type a name for the VM. Click **Next.**
   e. Associate the source network adapters in the OVF to networks defined on VMware. Click **Next.**
   f. A summary of the VM configuration is displayed. Click **Next.**
   g. The VMDK file for the OVF is now downloaded from the FTP site. When it is done, the Equalizer VM should now appear in your inventory.
3. If you downloaded and deployed the OVF file in the previous step, skip this step. Otherwise, if your client does not have Internet access, do the following:
   a. Download the *.ovf files from the EQOD part of the site, into the same directory on your local system:

      http://www.coyotepoint.com/content/eqos-10-support-page
   b. Select **File > Deploy OVF Template**.
   c. Click **Deploy from file**, browse to the directory to which you downloaded the files in the *.ovf* and *.vmdk* files, and select the *.ovf* file. Click **Next.**
   d. The OVF details are displayed. Click **Next.**
   e. Type a name for the VM. Click **Next.**

   f. Associate the source network adapters in the OVF to networks defined on VMware. Click **Next.**
   g. A summary of the VM configuration is displayed. Click **Next.**
   h. The VMDK file for the OVF is now downloaded from the local directory. When it is done, the EQOD VM should now appear in your inventory.

4. The first time you start Equalizer, login to the CLI on the VM console using the **touch** login (default password is **touch**). We recommend that you immediately change the default password for the **touch** login. Do this using the following CLI command:

```
eqcli > user touch password
```

Installing EQOD from a ZIP file

To install EQOD using the ZIP file distribution open the zip file that can be downloaded from:

http://www.coyotepoint.com/content/eqos-10-support-page

Follow the instructions for the VMware product you are using in the sections that follow.

VMware vSphere or vCenter Clients

VMware ESX and ESXi servers are managed using either the vSphere or vCenter management clients. For either of these products, the installation process is similar:

1. Copy the ZIP file onto the system running where vSphere or vCenter is installed. Unpack the ZIP file using utilities on that system.
2. Open the vSphere or vCenter client.
3. Select **Host > Configuration > Storage.**
4. Right-click on the DataStore on which you want to install EQOD and select **Browse.**
5. Select the **Upload** icon and then **Upload Folder.** Browse to the directory where you unpacked the ZIP file in Step 1, select the file **Equalizer.vmx**, and click **Open**.
6. After completion, open the new OnDemand directory in the DataStore Browser.
7. Right click on file **Equalizer.vmx** and select **Add to Inventory**.
8. Either accept the default VM name and resource pool or change them. Click **OK** to continue.
9. Once the virtual machine is loaded, EQOD should appear in the virtual machine list. Start (or play) the virtual machine by selecting the appropriate command from the menus or double-click on the virtual machine name.
10. EQOD will automatically boot and display the CLI login prompt. Login using the **touch** login (default password is **touch**).
11. We recommend that you immediately change the default password for the **touch** login. Do this using the following CLI command:

```
eqcli > user touch password
```

VMware Player and VMware Fusion

Besides running on dedicated hardware with the VMware ESX operating system, VMware can also run on Windows and MAC computers. VMware Workstation and VMware Player are Windows-based hypervisors, while VMware Fusion is the MAC version. After installing one of these products, follow these instructions to add the EQOD VM into either of these products.

1. Copy the ZIP file onto your Windows or MAC host. Unpack the ZIP file using utilities on your host system.
2. Open VM Player or Fusion and load the EQOD image:

    - On VM Player, select **File > Open a Virtual Machine**.
    - On VM Fusion, select **File > Open**.

    Browse to the location where you unpacked the ZIP file in Step 1, select the file EQOD**.vmx** and click **Open**.

3. Once the virtual machine is loaded, EQOD should appear in the virtual machine list. Start (or play) the virtual machine by selecting the appropriate command from the VMware menus or double-click on the virtual machine name.
4. EQOD will automatically boot and display the CLI login prompt. Login using the **touch** login (default password is **touch**).
5. We recommend that you immediately change the default password for the **touch** login. Do this using the following CLI command:

```
eqcli > user touch password
```

Licensing EQOD

When EQOD is first installed, it is unlicensed. In the unlicensed state, you can create objects but no clusters will accept connections until a valid license is installed. You can license your system offline using the CLI.

Before you can register, you will need:

Access to a new or existing Support Account. Information on how to create and manage a support account is provided in the Fortinet Support Portal User Guide. If your organization already has an account, obtain the user name and password information from your local account administrator to log in.

The serial number of the unit you want to register. You can find this information using either the CLI after installing the EQOD image on your VMware host:

1. Install EQOD on to your VMware host following the directions in the previous section.
2. Log in to the CLI using the **touch** login.
3. Enter the following CLI command:

```
eqcli > version
```

   Record the **System Serial Number** from the command output for later use in this procedure.

Once you have obtained both the login credentials of a support account and the System Serial Number of the unit to register, do the following:

1. Log in to https://support.fortinet.com using the login credentials obtained above.
2. Follow the instructions provided in the Registration Frequently Asked Questions under the heading “How do I register a Fortinet device?” to register your EQOD. When requested, enter the **System Serial Number** you obtained above into the appropriate form. Once registration is completed, the appliance serial number and other information will appear in the FortiCare Registration area.
3. Point to the **Asset** link at the top of the page and select **Manage/View Products** from the drop down list. A list of products that are currently registered will be displayed, including your EQOD.
4. Select the EQOD from the list of registered products and the **Product Info** screen will be displayed.
5. Click on the **License File Download** hyperlink to download the license file. Follow the prompts to download the file to an easily remembered location. It will have a ".lic" extension.

6. To upload the license using the CLI:

   a. Decompress the *.lic license file. It uses gzip compression and can be decompressed using any tool with gzip decompression functionality. If preferable, you can decompress the file using UNIX command line syntax:

```
dhcp-12:~ authorizeduser$ gzip -S .lic -d EQOD010000000498.lic
```

   When the license file decompresses a text file will be saved in it's place with the licensing information.

   b. Open the text file and highlightt the contents from the first bracket to the last as shown below. "Copy" to your computer's clipboard. (The display below is truncated)

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:lic="http://tempuri.org/lic.xsd"><SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<lic:getLicenseResponse>
<lic><item><item><name>license_version</name><value-str></value-str><value-
int>0</value-int></item>.......</lic:getLicenseResponse></SOAP-
ENV:Body></SOAP-ENV:Envelope>
```

c. Enter `eqcli >` **`license upload`** and "Paste" the license file contents at the end of the line and **ENTER**. The following is an example showing a license file.

```
eqcli > license upload license file contents
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:lic="http://tempuri.org/lic.xsd"><SOAP-ENV:Body SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<lic:getLicenseResponse>
<lic><item><item><name>license_version</name><value-str></value-str><value-
int>0</value-int></item>.......</lic:getLicenseResponse></SOAP-
ENV:Body></SOAP-ENV:Envelope>
eqcli: 12200287: Operation successful
eqcli >
```

d. Confirm that your EQOD is licensed by using the `eqcli >` **`version`** command. **`Support Information`** will be displayed at the bottom of the list if your EQOD has been properly licensed..

7. To upload the license using the GUI:

   a. Log in to the EQOD GUI using the **touch** login.

   b. Select the **System > Maintenance > Licensing** on the left navigational panel to display the licensing screen on the right.

   c. Click on the **Upload License** button and follow the prompts to select the license file that you saved in step 5.

   d. Click on **Commit** to upload the license file to EQOD.

   e. Confirm that your EQOD is licensed by clicking on **Global > Dashboard**on the left navigational pane. Your licensing information will be displayed within the **System Information** widget. In addition, select **System > Maintenance > Licensing** to display the uploaded license. Verify that the **Valid** line states **YES**.

Upgrading EQOD

EQOD can be upgraded to the latest release using the same methods in the CLI and GUI that are supported for hardware Equalizers. See the section "Upgrading to the Latest Release" on page 86 for instructions.

Appendix B

Using Certificates in HTTPS Clusters

Sections within this chapter include:

Using Certificates in HTTPS Clusters

The HTTPS protocol supports encrypted, secure communication between clients and servers. It requires that a Secure Sockets Layer (SSL) authentication handshake occur between a client and a server in order for a connection request to succeed.

When a client requests an HTTPS connection to a web server, the server (which has already been set up to support SSL connections) sends a server certificate to the client for verification. The client checks the content of the certificate against a local database of Certificate Authorities, and if it finds a match the connection is made. If no match is found (as is often the case with self-signed certificates), the browser will display a warning and ask if you want to continue with the connection.

A further level of trust can be enabled by setting the server up to request a client certificate in addition to the server certificate. Copies of the client certificate are pre-installed on both client and server. When the server sends the server certificate to the client, it also sends a request for a certificate from the client. Once the client accepts the server certificate as described above, it sends the client certificate to the server for verification. The server compares the client certificate it receives with its local copy of the client certificate, and if they match the connection is made.

Each Layer 7 HTTPS cluster requires a server certificate; client certificates are optional.

Web servers (such as Apache) and browsers (such as Internet Explorer and Firefox) are delivered with pre-installed Trusted Root Certificates. Trusted Root Certificates are used to validate the server and client certificates that are exchanged when an HTTPS connection is established.

Equalizer supports self-signed certificates, as well as signed certificates from Trusted Root Certificate Authorities and from Certificate Authorities (CAs) without their own Trusted Root CA certificates. If a CA without its own Trusted Root CA certificate issues your certificate, you will need to install at least two certificates: a server certificate and a chained root (or intermediate) certificate for the CA. The intermediate certificate associates the server certificate with a Trusted Root certificate.

About Server Certificates

In a typical HTTPS scenario described above, the client and server are communicating directly, and the server is doing all the work of encrypting and decrypting packets, and sending the server certificate to the client. If you have many systems servicing requests for the same website, you need to install certificates on each server.

With Equalizer, you do not need to install a server certificate on every server in a Layer 7 HTTPS cluster. Since certificates are associated with host names and not IP addresses, you only need a server certificate for each HTTPS cluster and the certificates are installed only on Equalizer-- not on each server. This reduces maintenance by reducing the number of certificates required for a group of systems serving content for the same host name.

When a client requests a connection to an HTTPS cluster, Equalizer establishes the HTTPS connection with the client, off loading SSL processing from all the servers in the HTTPS cluster. Equalizer communicates with the clients via HTTPS; the traffic between Equalizer and the servers in an HTTPS cluster is HTTP (i.e., unencrypted).

Compared to the typical scenario where each client is establishing direct HTTPS connections with servers, encrypting and decrypting packets, and serving content as well, SSL offloading improves the overall performance of the cluster.

For even better performance, some Equalizer models are equipped with SSL Hardware Acceleration. With hardware acceleration, processing for cipher suites supported by acceleration hardware is done by dedicated hardware, enhancing overall HTTPS throughput.

Note that HTTPS and certificates can also be used on servers in Layer 4 TCP and UDP clusters, but you will need to install a server and client certificate on each server in the cluster (since Equalizer is not doing any HTTPS/SSL processing in Layer 4). In this scenario, no certificates are installed on Equalizer.

About Client Certificates

If you want to use client certificates with an HTTPS cluster, you’ll need to get a signed client certificate from a CA, or create a self-signed certificate. A client certificate needs to be installed on each client that will access the Equalizer cluster, as well as on Equalizer.

Just as with server certificates, you may need to install a client certificate and a chained root certificate, if you obtain your certificates from a CA without its own Trusted Root CA certificate. Some sites prefer to use self-signed certificates for clients, or set up their own local CA to issue client certificates.

Client certificates can be used in two ways with Equalizer:

1. Install the entire client certificate chain on Equalizer. This requires that every client passes the exact same certificate to Equalizer for validation.
2. Install an intermediate CA certificate as the client certificate on Equalizer. This allows unique certificates to be used on clients and a single client certificate to be uploaded to Equalizer. Following this method requires some certificate processing on the servers behind Equalizer in order to prevent access by clients with revoked certificates. This method, therefore, should be used only under the following conditions:

    a. If the site is able to use an intermediate CA, or multiple CAs, which signs all and only certificates authorized for use with the cluster.
    AND

    b. If the application running on the servers behind Equalizer is able to perform Certificate Revocation List (CRL) processing by matching the CSN (certificate serial number) to the intermediate CA's CRL, and does so for all requests,
    THEN

c. Equalizer can safely support the use of individual client certificates for different clients, by appropriately setting the verify depth option for the HTTPS cluster and uploading the intermediate CA's certificate to the cluster as the client certificate. If client certificates use different CAs, multiple intermediate CAs can be uploaded to Equalizer in a single file.

This method ensures that only certificates that pass the CRL check on the server can be used to access the cluster. Note that this method also assumes that validating the intermediate certificate only in (b) above is sufficiently secure for the site.

General Certificate Guidelines

Currently, the following certificate/key file formats are supported:

1. **PEM** - PEM format certificates/keys are ascii files that usually use a ".pem" extension with the file name. PEM stands for Privacy Enhanced Mail. A PEM-format certificate contains a Base64 encoded DER certificate, enclosed between **"-----BEGIN CERTIFICATE-----"** and **"-----END CERTIFICATE-----"** tokens. Keys encoded using the PEM format would have **"-----BEGIN PRIVATE KEY-----**" and **"-----END PRIVATE KEY-----"** tokens. PEM format certificates/keys only are supported using the GUI.
2. **PKCS #12** - PKCS #12 format files are binary files, usually with a ".p12’"extension with the file name.
3. **PFX** - PFX format files are also in PKCS #12 format, however, with additional Microsoft specifics. These files usually have a ".pfx" extension with the file name. PFX files are supported in the GUI, however, not in the CLI.

Currently, PEM-format certificates and keys must be uploaded separately in the CLI using the certfile and keyfile parameters in the certificate context or as shown below in the GUI.

PKCS #12 and PFX format files usually contain both the certificate and the associated key. You can upload this file once as either the certfile or the keyfile in the GUI. The GUI will separate the keyfile and the certfile behind the scenes and store them appropriately. You can also upload the same file as both the certfile and the keyfile.

If you have uploaded a certificate that doesn't match the cipher suite that is configured for the HTTPS cluster, you will no longer be able to log into the GUI. You will need to supply the correct certificate/key pairing. In the meantime, you can enable HTTP access to the GUI temporarily to enter the proper certificate/key pairing to enable HTTPS access.

Software vs. Hardware Encryption/Decryption

Without hardware SSL acceleration, all Layer 7 HTTPS encryption and decryption is performed by software, using Equalizer's CPU and memory. With hardware acceleration, all SSL operations for Layer 7 HTTPS clusters are performed on dedicated hardware, thus offloading both the servers behind Equalizer and Equalizer itself -- freeing more resources for traffic and application management.

In terms of configuration, both software and hardware SSL operations require a list of cipher suites (encryption algorithms) to be used to encrypt and decrypt HTTPS traffic.

The following table indicates the encryption support for specific models.

Platform	**SSL offloading**
E250GX	Software only
E370LX E350GX	Software only
E470LX E450GX	Hardware and Software
E670LX E650GX	Hardware and Software
E970LX	Hardware and Software
OnDemand	Software only

Configuring Cipher Suites

A cipher is an algorithm that performs encryption or decryption. It transforms plain text into a coded set of data (cipher text) that is not reversible without a key. For example, AES and DES are examples of secret key block ciphers. The complete encryption algorithm is the cipher plus the technique used to apply the cipher to the message, which can be a series of steps.

he **Configured Cipher List** (`cipherspec` in the CLI)for an HTTPS cluster lists all of the ciphers that can be negotiated between Equalizer and an incoming client attempting to connect to an HTTPS cluster. Similarly, the client application will have its own list of ciphers that it supports. The client and Equalizer need to go through a process of negotiating the cipher that will be used for the client connection -- if they cannot find a match, the connection will fail. The process of negotiating a cipher for a client connection is as follows:

1. During the SSL handshake phase of the connection, the client sends Equalizer a list of the ciphers it supports.
2. Equalizer examines the client cipher list in the order it is specified, chooses the first cipher that matches a cipher specified in the cluster’s cipher suite parameter, and responds to the client. If none of the ciphers offered by the client are in the cipher suite list for the cluster, the SSL handshake fails.

It is therefore vital that you ensure that there is at least one match between the list of ciphers supported by clients connecting to an HTTPS cluster and the cipher suite list for the cluster.

The **Configured Cipher List (`cipherspec`)** HTTPS cluster parameter lists the supported encryption algorithms for incoming HTTPS requests. If a client request comes into Equalizer that does not use a cipher in this list, the connection is refused. If this field is blank, then any cipher suite supported by Equalizer’s SSL implementation (or by Hardware SSL Acceleration, when enabled) will be accepted.

To view or set the **Configured Cipher List** for a cluster on the GUI, click on the cluster name in the left navigational pane, select the HTTPS cluster, and then select the **Security > SSL tab** in the right pane to display the Cluster Security SSL screen. (See "Layer 7 SSL Security (HTTPS Clusters)" on page 384)

To view or set the cipher suite for an HTTPS cluster on the CLI enter

```
eqcli > show cluster clustername.
```

For example:

```
eqcli > show cluster cluster_https

L7 Cluster Name            : cluster_https
Protocol                   : https
IP Address                 : 2.5.8.11
Port                       : 53
Preferred Peer             :
VID                        : unassigned
Client Timeout             : 10
Server Timeout             : 60
Connection Timeout         : 10
Sticky Timeout             : 0
Sticky Netmask             : 32
Custom Header              :
CRL                        :
CA Certificate             :
Cipher Spec                : AES128-SHA:RC4-SHA:RC4-MD5:AES256-SHA:!SSLv2
Validation Depth           : 9
Flags                      : allow_utf8, allow_sslv3, ignore_critical_
extns, allow_tls10, rewrite_redirects, insert_client_ip
Default Certificate        :
SNI Certificate Objects    :
Server Pool                :
Responder                  :
Cookie Path                :
Cookie Domain              :
Cookie Age                 : 0
Cookie Generation          : 0
Persist Type               : coyote_cookie_2
Minimum Compression        : 1024
Compression mime_type      :
text/*:application/msword:application/postscript:application/rtf:applicati-
on/x-csh:application/x-javascript:application/x-sh:application/x-
shar:application/x-tar:application/x-
tcl:application/xslt+xml:audio/midi:audio/32kadpcm:audio/x-
wav:image/bmp:image/tiff:image/x-rgb
Config Header Edit Script  :
eqcli cl-clu*> 12000004: You have 5 pending alert notifications.
```

Default Cipher Suites

On both software only and hardware accelerated systems, the default cipher suite setting is:

AES128-SHA:DES-CBC3-SHA:RC4-SHA:RC4-MD5:AES256-SHA:!SSLv2

See "Replacing the Default Certificate, Key, and Cipherspec" on page 254 for descriptions on replacing the default cipher suite.

Updating the Cipher Suites Field

This field can be used to specify a custom cipher suite required by the servers in a cluster. In general, to add a cipher suite, you specify a plus sigh (+) and then the name of the suite. To specifically exclude a cipher suite, use an exclamation point (!).

For example, SSLv2 encryption is supported by default. If your servers are required to support medium and high encryption using SSLv3 only, you can add "SSLv3"to the cipher suite. For example, the following cipher suite string will cause all non-SSLv3 client requests to be refused:

AES128-SHA:DES-CBC3-SHA:RC4-SHA:RC4-MD5:AES256-SHA:!SSLv2:+SSLv3

The **Cipher Suites** field requires a string in the format described in the OpenSSL cipher suite documentation, at:

```
http://www.openssl.org/docs/apps/ciphers.html
```

To update in the GUI, refer to "Layer 7 SSL Security (HTTPS Clusters)" on page 384

To update in the CLI, enter ciphers as in the format:

```
eqcli > cluster clustername cipherspec cipher
```

For example:

```
eqcli cl-clu*> cipherspec DES-CBC-SHA

eqcli cl-clu*> commit
eqcli: 12000287: Operation successful

eqcli cl-clu*> show
L7 Cluster Name             : cluster_https
Protocol                    : https
IP Address                  : 2.5.8.11
Port                        : 53
Preferred Peer              :
VID                         : unassigned
Client Timeout              : 10
Server Timeout              : 60
Connection Timeout          : 10
Sticky Timeout              : 0
Sticky Netmask              : 32
Custom Header               :
CRL                         :
```

```
CA Certificate              :
Cipher Spec                 : DES-CBC-SHA
Validation Depth            : 9
Flags                       : allow_utf8, allow_sslv3, ignore_critical_
extns, allow_tls10, rewrite_redirects, insert_client_ip
Default Certificate         :
SNI Certificate Objects     :
Server Pool                 :
Responder                   :
Cookie Path                 :
Cookie Domain               :
Cookie Age                  : 0
Cookie Generation           : 0
Persist Type                : coyote_cookie_2
Minimum Compression         : 1024
Compression mime_type       :
text/*:application/msword:application/postscript:application/rtf:applicati-
on/x-csh:application/x-javascript:application/x-sh:application/x-
shar:application/x-tar:application/x-
tcl:application/xslt+xml:audio/midi:audio/32kadpcm:audio/x-
wav:image/bmp:image/tiff:image/x-rgb
Config Header Edit Script   :
eqcli cl-clu*>
```

Supported Software Cipher Suites

The following table lists the **software** cipher suites supported by Equalizer.

Ciphers	Key Exchange	Protocol	Authentication	Encryption	Message Authentication Code
AES256-GCM-SHA384	RSA	TLSv1.2	RSA	AESGCM(256)	AEAD
AES256-SHA256	RSA	TLSv1.2	RSA	AES(256)	SHA256
AES256-SHA	RSA	SSLv3	RSA	AES(256)	SHA1
CAMELLIA256-SHA	RSA	SSLv3	RSA	Camellia(256)	SHA1
DES-CBC3-SHA	RSA	SSLv3	RSA	3DES(168)	SHA1
AES128-GCM-SHA256	RSA	TLSv1.2	RSA	AESGCM(128)	AEAD
AES128-SHA256	RSA	TLSv1.2	RSA	AES(128)	SHA256
AES128-SHA	RSA	SSLv3	RSA	AES(128)	SHA1
SEED-SHA	RSA	SSLv3	RSA	SEED(128)	SHA1
CAMELLIA128-SHA	RSA	SSLv3	RSA	Camellia(128)	SHA1
IDEA-CBC-SHA	RSA	SSLv3	RSA	IDEA(128)	SHA1
RC4-SHA	RSA	SSLv3	RSA	RC4(128)	SHA1
RC4-MD5	RSA	SSLv3	RSA	RC4(128)	MD5
DES-CBC-SHA	RSA	SSLv3	RSA	DES(56)	SHA1
EXP-DES-CBC-SHA	RSA(512)	SSLv3	RSA	DES(40)	SHA1 export
EXP-RC2-CBC-MD5	RSA(512)	SSLv3	RSA	RC2(40)	MD5 export
EXP-RC4-MD5	RSA(512)	SSLv3	RSA	RC4(40)	MD5 export
AES256-GCM-SHA384	RSA	TLSv1.2	RSA	AESGCM(256)	AEAD

Supported PFS Ciphersuites

The following table lists the PFS ciphersuites supported by Equalizer.

Ciphers	Key Exhange
ECDHE-RSA-AES256-GCM-SHA384	RSA
ECDHE-ECDSA-AES256-GCM-SHA384	ECDSA
ECDHE-RSA-AES256-SHA384	RSA
ECDHE-ECDSA-AES256-SHA384	ECDSA
ECDHE-RSA-AES256-SHA	RSA
ECDHE-ECDSA-AES256-SHA	ECDSA
DHE-DSS-AES256-GCM-SHA384	DSS
DHE-RSA-AES256-GCM-SHA384	RSA
DHE-RSA-AES256-SHA256	RSA
DHE-DSS-AES256-SHA256	DSS
DHE-RSA-AES256-SHA	RSA
DHE-DSS-AES256-SHA	DSS
DHE-RSA-CAMELLIA256-SHA	RSA
DHE-DSS-CAMELLIA256-SHA	DSS
ECDHE-RSA-AES128-GCM-SHA256	RSA
ECDHE-ECDSA-AES128-GCM-SHA256	ECDA
ECDHE-RSA-AES128-SHA256	RSA
ECDHE-ECDSA-AES128-SHA256	ECDSA
ECDHE-RSA-AES128-SHA	RSA
ECDHE-ECDSA-AES128-SHA	ECDSA
DHE-DSS-AES128-GCM-SHA256	DSS
DHE-RSA-AES128-GCM-SHA256	RSA
DHE-RSA-AES128-SHA256	RSA
DHE-DSS-AES128-SHA256	DSS
DHE-RSA-AES128-SHA	RSA
DHE-DSS-AES128-SHA	DSS

Ciphers	Key Exhange
DHE-RSA-SEED-SHA	RSA
DHE-DSS-SEED-SHA	DSS
DHE-RSA-CAMELLIA128-SHA	RSA
DHE-DSS-CAMELLIA128-SHA	DSS
ECDHE-RSA-RC4-SHA	RSA
ECDHE-ECDSA-RC4-SHA	ECDSA
ECDHE-RSA-DES-CBC3-SHA	RSA
ECDHE-ECDSA-DES-CBC3-SHA	ECDSA

Supported Hardware Cipher Suites

Ciphers	Protocol	Key Exchange	Authentication	Encryption	Message Authentication Code
TLS1_RSA_EXPORT1024_WITH_ RC4_56_MD5	<=TLS 1.1	RSA	RSA	RC4	MD5 export
TLS1_RSA_EXPORT1024_WITH_ RC4_56_SHA;	<=TLS 1.1	RSA	RSA	RC4	SHA export
SSL3_RSA_RC4_40_MD5	<=TLS 1.1	RSA	RSA	RC4 (40)	MD5
TLS1_RSA_EXPORT1024_WITH_ DES_CBC_SHA	<=TLS 1.1	RSA	RSA	DES	SHA
SSL3_RSA_DES_40_CBC_SHA;	<=TLS 1.1	RSA	RSA	DES (40)	SHA
SSL3_RSA_DES_64_CBC_SHA	<=TLS 1.1	RSA	RSA	DES (64)	SHA
SSL3_RSA_RC4_128_MD5	SSL 3.0	RSA	RSA	RC4 (128)	SHA
SSL3_RSA_RC4_128_SHA	SSL 3.0	RSA	RSA	RC4 (128)	SHA
SSL3_RSA_DES_192_CBC3_SHA	SSL 3.0	RSA	RSA	DES (192)	MD5
SSL2_RC4_128_WITH_MD5	SSL 2.0	RSA	RSA	RC4 (128)	MD5
SSL2_RC4_128_EXPORT40_ WITH_MD5	SSL 2.0	RSA	RSA	RC4 (128)	MD5
SSL2_DES_64_CBC_WITH_MD5	SSL 2.0	RSA	RSA	DES (64)	MD5
SSL2_DES_192_EDE3_CBC_ WITH_MD5	SSL 2.0	RSA	RSA	DES (192)	MD5
TLS1_RSA_WITH_AES_128_SHA	TLS 1.0	RSA	RSA	AES (128)	SHA
TLS_RSA_WITH_AES_256_SHA	TLS 1.0	RSA	RSA	AES (256)	SHA
TLS_RSA_WITH_AES_128_ SHA256	TLS 1.2	RSA	RSA	AES (128)	SHA (256)
TLS_RSA_WITH_AES_256_ SHA256	TLS 1.2	RSA	RSA	AES (256)	SHA (256)
TLS1_RSA_WITH_AES_128_ GCM_SHA256	TLS 1.2	RSA	RSA	AESGCM (128)	SHA (256)
TLS1_RSA_WITH_AES_256_ GCM_SHA384	TLS 1.2	RSA	RSA	AESGCM (256)	SHA (384)

Enabling HTTPS with a Server Certificate

The following are the steps to follow to obtain and install a server certificate, and verify that it works.

1. Generate a Server Certificate Signing Request or a Self-Signed Server Certificate.

   To get a server certificate, do one of the following:

   a. **Create a Certificate Signing Request (CSR) and send it to a Certificate Authority for signing**. This provides the highest level of trust to the client, as the client can be assured that the certificate it receives from the server (in this case, Equalizer) was approved (i.e., digitally signed) by a trusted third party. Thus, the client has the assurance of a third party that the server to which it is connecting is identifying itself legitimately (and is not impersonating the legitimate server's identity). See "Generating a CSR and Getting It Signed by a CA" on page 921.

   b. **Create a certificate and sign it yourself**. This provides a lower level of trust, since the client is essentially trusting the server to identify itself. Self-signed certificates are relatively easy to counterfeit, and are only recommended for use on internal, non-production, or test configurations. See "Generating a Self-Signed Certificate" on page 923.

2. Create the HTTPS cluster.

   When creating an HTTPS cluster, the default flags and parameters are acceptable for most server certificate configurations.

   For more information on SSL parameters, see the section "Layer 7 SSL Security (HTTPS Clusters)" on page 384.

3. Install the Server Certificate on Equalizer. See the section "Layer 7 Security Certificate Screen (HTTPS Clusters)" on page 380.
4. Try connecting to the Cluster via HTTPS.

   From a client browser, open **https://cluster**, where **cluster** is the network node name or IP address of the HTTPS cluster. The browser may notify you that it is accepting a certificate from the server and ask for confirmation.

   Once you accept the certificate, the requested page should be displayed.

Enabling HTTPS with Server and Client Certificates

The following are the steps to follow to obtain and install both server and client certificates, and verify that they work.

1. Perform the procedure in the previous section to enable HTTPS with a server side certificate.
2. Generate a Client Certificate Signing Request or a Self-Signed Client Certificate.

   In Step 1, you created a server certificate. Now, follow the same procedure to generate a client certificate; do one of the following:

   a. Create a Certificate Signing Request (CSR) and send it to a Certificate Authority for signing. See "Generating a CSR and Getting It Signed by a CA" on page 921.

   b. Create a certificate and sign it yourself. see "Generating a Self-Signed Certificate" on page 923.

   Many organizations choose to use third-party signed certificates for their HTTPS clusters, and use self-signed certificates for their clients.

3. Modify the HTTPS cluster to request a client certificate.

   a. Select the HTTPS cluster in the left navigational pane on the GUI and then select the SSL tab in the right pane.

   b. Enable the **Require Client Certificate** flag; this tells Equalizer to request a client certificate when a client attempts to connect to this cluster.

   c. By default, the client certificate verification depth is set to 2. This number indicates the number of levels in a certificate chain that the Equalizer will process before stopping verification. This default depth may need to be raised if you received more than one chained root certificate in addition to a client certificate from your Certificate Authority. Note that this setting has an impact on performance, since SSL operations are resource-intensive.

   d. By default, Equalizer requests a client certificate, but does not require the client to provide one. Enable the require certificate flag to require that a client return a valid certificate before connecting.

   e. By default, the client's certificate will be re-validated if the SSL connection needs to be renegotiated. (Renegotiation is a feature of SSL, can occur for any of a number of reasons, and may be initiated by Equalizer or the client browser.) Enable the verify once flag to tell Equalizer not to re-evaluate the client certificate even if SSL renegotiation occurs. This can have a positive performance impact if many SSL renegotiation is occurring during normal operations.

   f. Select **Commit** to save your changes to the cluster definition.

   For more information on SSL parameters, see the section"Layer 7 SSL Security (HTTPS Clusters)" on page 384

4. Install the Client Certificate on Equalizer. Use the Equalizer Administration Interface to install the client certificate. See "Layer 7 SSL Security (HTTPS Clusters)" on page 384
5. Install the Client Certificate on all clients.
   Import the client certificate into the client browser’s list of certificates.Follow the instructions for Mozilla Firefox, Internet Explorer, Chrome, etc.
6. Try connecting to the Cluster via HTTPS.
   From a client browser, open **https://cluster**, where cluster is the network node name or IP address of the HTTPS cluster. The browser may notify you that it is accepting a certificate from the server and ask for confirmation. Once you accept the certificate, the server should ask for a client certificate; your browser may ask you to choose one. After the client certificate is sent to the server and accepted, the requested page should be displayed.

Generating a CSR and Getting It Signed by a CA

Most CA vendors provide a means of generating a Certificate Signing Request (CSR) on their websites, and we recommend that you use the CA website to generate the CSR. For several good tutorials on how to get your certificates signed, please see:

http://sial.org/howto/openssl/

A CSR can also be generated using the OpenSSL tools on any system, including Windows. The examples below were executed on a Windows system with the OpenSSL tools installed.

Note that only the most basic **openssl** command options are shown in these examples. See the **openssl**(1) and **req**(1) manual pages for the SSL implementation on your system for more information.

Note - Generating a CSR on Equalizer is NOT supported. Consult the Certificate Authority that supplies your SSL certificates and use the tools that they recommend.

Generating a CSR using OpenSSL

1. Navigate to an appropriate directory on your system, and create a new directory to hold your CSR, certificate, and private key.
2. Generate the CSR by entering this command:

   ```
   openssl req -new -newkey rsa:1024 -out cert.csr
   ```

   This begins an interactive session to generate a CSR, and also generates a new private key to be output into a file named privkey.pem. If you already have a private key, use -key filename (instead of -newkey rsa:1024) to specify the file containing the private key.

   It is recommended that you do not share your private key. Anyone who obtains your private key will be able to use it to decrypt cluster traffic, so keep your private keys in a secure location.

   After generating the private key, the following prompts are displayed (example responses shown):

   ```
   Enter PEM pass phrase: <password>
   Verifying - Enter PEM pass phrase: <password>
   Country Name (2 letter code) [AU]:US
   State or Province Name (full name) [Some-State]:New York
   Locality Name (eg, city) []:Millerton
   Organization Name (eg, company) [Internet Widgits Pty Ltd]:CPS Inc.
   Organizational Unit Name (eg, section) []:Engineering
   Common Name (eg, YOUR name) []:mycluster.example.com
   Email Address []:admin@example.com
   ```

Make sure you remember the **password** you specify, as you will need it to install and use the certificate.

For a server certificate, the **Common Name** provided must be the DNS-resolvable fully qualified domain name (FQDN) used by the cluster. When a client receives the certificate from the server, the client browser will display a warning if the **Common Name** does not match the hostname of the request URI.

For a client certificate, the **Common Name** in the client’s copy of the certificate is only compared to the Common Name in the copy of the client certificate on the server, so **Common Name** can be any value.

3. Visit the website of an SSL Certificate Authority (CA) to submit the `cert.csr` file to the CA.
4. Once the CA returns your signed certificate (usually in email), go to "Generating a Self-Signed Certificate" on page 923 for more information.

Generating a CSR and Installing a Certificate on Windows Using IIS

Using Internet Information Services (IIS) is optional when creating and managing certificates for Equalizer Layer 7 HTTPS clusters and clients. In fact, one of the advantages of using Equalizer is that only one server certificate is required for an HTTPS cluster. The cluster certificate is installed on Equalizer, not on the servers in the HTTPS cluster. So, you do not need to use IIS on each server to create and install certificates. This reduces the amount of effort spent administering server certificates.

For Layer 4 TCP and UDP clusters, certificates are not installed on Equalizer, and you will need to install a server certificate on each server in the cluster (since Equalizer is not doing any HTTPS/SSL processing in Layer 4).

Please refer to the IIS documentation from Microsoft.for descriptions for generating a CSR and installing a signed certificate on Windows using IIS.

Generating a Self-Signed Certificate

To generate a self signed certificate in PEM format:

1. Generate a self-signed x509 format certificate by entering this command:

   ```
   openssl req -new -x509 -newkey rsa:1024 -out selfcert.pem -days 1095
   ```

   This creates a self-signed certificate (*selfcert.pem*) that will be valid for 1095 days (about three years) andalso generates a new private key to be output into a file named **privkey.pem**. If you already have a private key, use **- key *filename*** instead of **-newkey rsa:1024** to specify the file containing the private key.

   After generating the private key, the following prompts are displayed (example responses shown):

   ```
   Enter PEM pass phrase: <password>
   Verifying - Enter PEM pass phrase: <password>
   Country Name (2 letter code) [AU]:US
   State or Province Name (full name) [Some-State]:New York
   Locality Name (eg, city) []:Millerton
   Organization Name (eg, company) [Internet Widgits Pty Ltd]:CPS Inc.
   Organizational Unit Name (eg, section) []:Engineering Common Name (eg, YOUR name)
   []:myclient.example.com
   Email Address []:admin@example.com
   ```

   Depending on the tool you use to create the certificate, you may also be asked for a challenge password and other optional information. Make sure you remember the **password** (and, if prompted, the challenge password) you specify, as you will need it to install the certificate.

   The **Common Name** provided must be the DNS-resolvable fully qualified domain name (FQDN) used by the cluster. For a server certificate, when the client receives the certificate from the server, the browser will display a warning if the **Common Name** does not match the hostname of the request URI. For a client certificate, the Common Name in the client's copy of the certificate is only compared to the **Common Name** in the copy on the server, so this can be any value.

2. Combine the private key and certificate into one file, using a command like the following:

   ```
   cat selfcert.pem privkey.pem > clustercert.pem
   ```

3. You can now install your self signed certificate and private key file, `clustercert.pem`, on Equalizer and your clients, as appropriate.

Installing Certificates for an HTTPS Cluster

Refer to "Layer 7 Security Certificate Screen (HTTPS Clusters)" on page 380 for a description of installing certificates on an HTTPS cluster.

Converting a Certificate from PEM to PKCS12 Format

Many browsers, such as FireFox and Internet Explorer, require private keys and certificates in PKCS12 format for installation. In order to install client and intermediate certificates into these browsers, you will first have to convert them from PEM format to PKCS12 format. (Note: if you created your certificate using IIS as explained in the previous section, then your certificate is already in PKCS12 format; it can be installed directly into a browser without conversion.)

Like PEM format, PKCS12 format supports having all your certificates and your private key in one file. If you created the file `clientprivcert.pem` (containing the client certificate, the private key, and any intermediate certificates), then converting the file to PKCS12 is simple:

```
openssl pkcs12 -export -in clientprivcert.pem -out clientprivcert.pfx
```

The resulting file, `clientprivcert.pfx`, can now be installed into all client browsers that will be accessing the cluster that requires a client certificate.

Refer to the Microsoft, Mozilla, Google, and Apple for documentation for instructions on installing certificates on IE, Firefox, Chrome and Safari, respectively.

Appendix C

Using the File Editor

Sections within this chapter include:

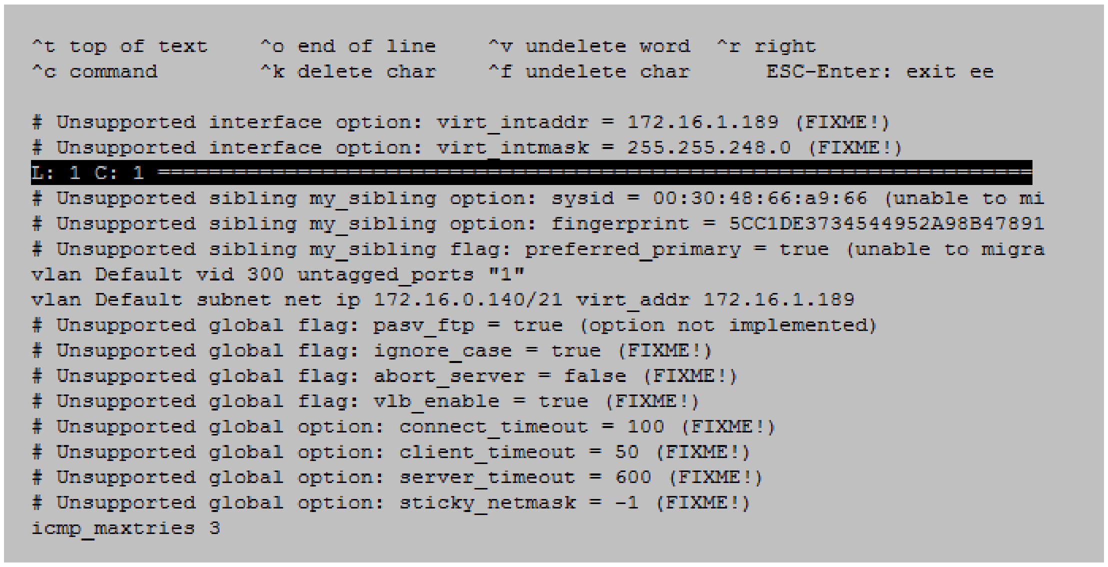
Editing Files

Files from the data store, for example, can be edited using the **`files edit`** command in the CLI using the "ee" editor . The most common example for using this feature is to edit CLI scripts which can then be executed using the **`run_script`** command, but there are other uses as well. You will be able to edit existing files, however you will not be able to create and save new files to the data store. (Use the files download command to place a new file into the datastore).

Examples of files that can be edited are **`responder html edit`**, **`certificate certfile/keyfile edit`**, **`files edit`**, and others. In the example below the files edit command is used. In the example below a configuration backup file script is opened for editing:

```
eqcli > files edit backup script
```

When this command is executed, the "*ee*" editor will be displayed:



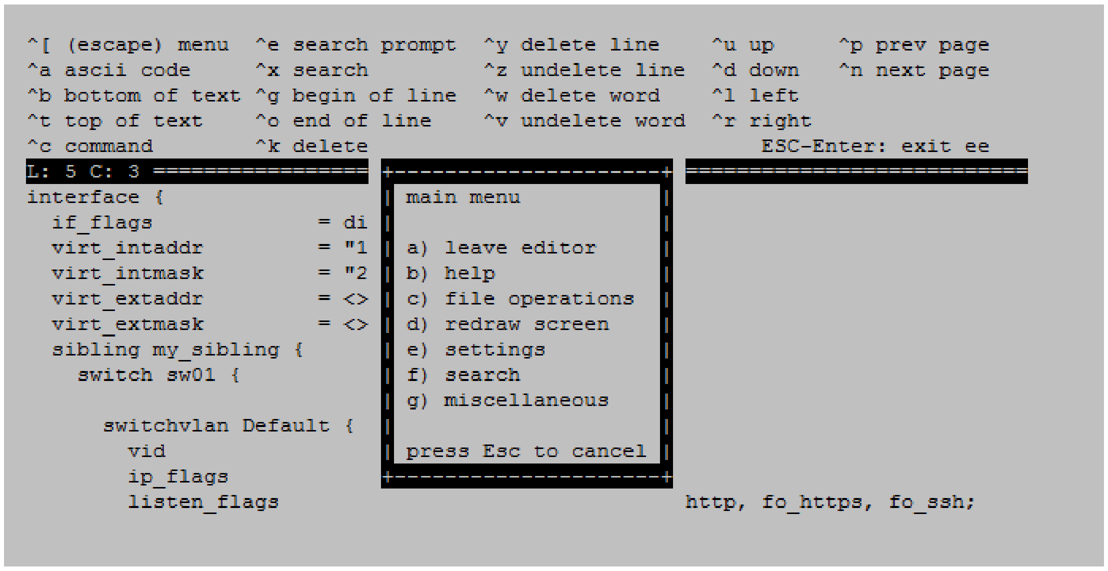
```
^t top of text    ^o end of line    ^v undelete word  ^r right
^c command        ^k delete char    ^f undelete char      ESC-Enter: exit ee

# Unsupported interface option: virt_intaddr = 172.16.1.189 (FIXME!)
# Unsupported interface option: virt_intmask = 255.255.248.0 (FIXME!)
L: 1 C: 1 ===============================================================
# Unsupported sibling my_sibling option: sysid = 00:30:48:66:a9:66 (unable to mi
# Unsupported sibling my_sibling option: fingerprint = 5CC1DE3734544952A98B47891
# Unsupported sibling my_sibling flag: preferred_primary = true (unable to migra
vlan Default vid 300 untagged_ports "1"
vlan Default subnet net ip 172.16.0.140/21 virt_addr 172.16.1.189
# Unsupported global flag: pasv_ftp = true (option not implemented)
# Unsupported global flag: ignore_case = true (FIXME!)
# Unsupported global flag: abort_server = false (FIXME!)
# Unsupported global flag: vlb_enable = true (FIXME!)
# Unsupported global option: connect_timeout = 100 (FIXME!)
# Unsupported global option: client_timeout = 50 (FIXME!)
# Unsupported global option: server_timeout = 600 (FIXME!)
# Unsupported global option: sticky_netmask = -1 (FIXME!)
icmp_maxtries 3
```

A list of commands is located at the top of the display.

The present location of your cursor is displayed in a highlighted block. For example **`L: 1 C: 1`** indicates that your cursor is on line 1, column 1 within the file contents. **`L:3 C;5`** indicates that your cursor is on line 3, column 5 within the file contents, etc.

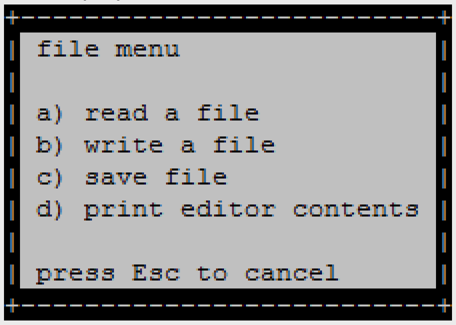
Use your keyboard's arrow keys (← ↑ ↓ →) to navigate to the text that requires editing.

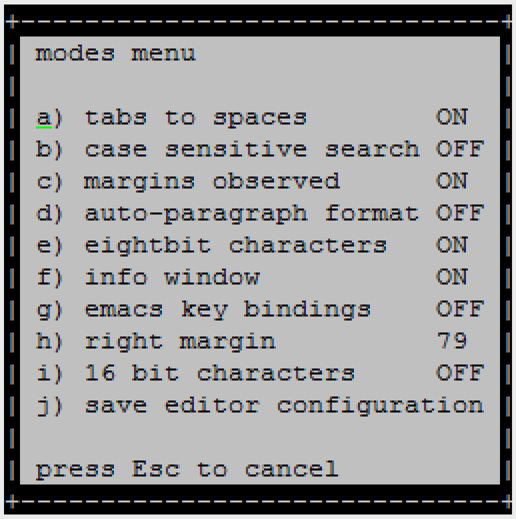
When you have finished editing enter **`ESC`** and a menu will be displayed as follows:

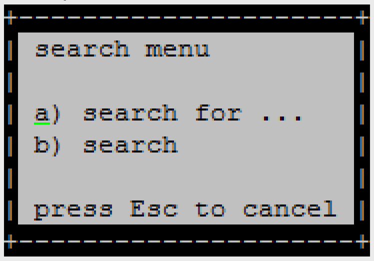


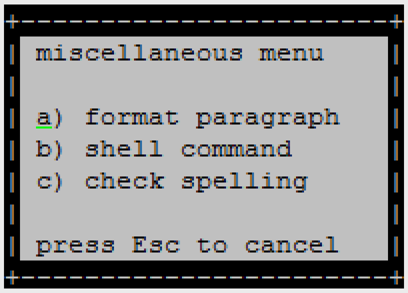
```
^[ (escape) menu  ^e search prompt  ^y delete line    ^u up      ^p prev page
^a ascii code     ^x search         ^z undelete line  ^d down    ^n next page
^b bottom of text ^g begin of line  ^w delete word    ^l left
^t top of text    ^o end of line    ^v undelete word  ^r right
^c command        ^k delete                                ESC-Enter: exit ee
L: 5 C: 3 ================= +--------------------+ ==========================
interface {                 | main menu          |
  if_flags             = di |                    |
  virt_intaddr         = "1 | a) leave editor    |
  virt_intmask         = "2 | b) help            |
  virt_extaddr         = <> | c) file operations |
  virt_extmask         = <> | d) redraw screen   |
  sibling my_sibling {      | e) settings        |
    switch sw01 {           | f) search          |
                            | g) miscellaneous   |
      switchvlan Default {  |                    |
        vid                 | press Esc to cancel|
        ip_flags            +--------------------+
        listen_flags                               http, fo_https, fo_ssh;
```

Main and Submenu Commands

a) leave editor	Leaves the ee editor. You will be prompted to save changes before exiting.
b) help	Will display a complete list of Control Keys and Commands.
c) file operations	Will display a submenu of commands that includes:  `file menu` `a) read a file` `b) write a file` `c) save file` `d) print editor contents` `press Esc to cancel` **read a file**,**write afile**and **print editor contents** are all restricted and not available **save file** - will save the changes that you made to the file.
d) redraw screen	Will redraw the open screen.

e) settings	Will display the following **modes menu** 
	```
	modes menu

	a) tabs to spaces          ON
	b) case sensitive search OFF
	c) margins observed       ON
	d) auto-paragraph format OFF
	e) eightbit characters    ON
	f) info window            ON
	g) emacs key bindings     OFF
	h) right margin           79
	i) 16 bit characters      OFF
	j) save editor configuration

	press Esc to cancel
	```

f) search	Will open a search submenu with 2 options: 
	```
	search menu

	a) search for ...
	b) search

	press Esc to cancel
	```
	 a) search for - will prompt you to enter a search term(s)
	b) search - [not available]

g) miscellaneous	Will display the following **miscellaneous menu**: 
	```
	miscellaneous menu

	a) format paragraph
	b) shell command
	c) check spelling

	press Esc to cancel
	```

Appendix D
Port Numbers

Sections within this chapter include:

Port Numbers

Communications between the Equalizer appliance, clients, and servers requires that any routers and firewalls between them permit specific protocols and port numbers.

Default Ports Used by Equalizer for Outgoing Traffic (Client)

Port Number	IP Protocol Number/ Service	Purpose
N/A	ARP	HA failover of network interfaces.
N/A	ICMP	• Server health checks. • Execute ping and execute traceroute
25	TCP	SMTP for alert email.
53	UDP	Envoy DNS queries.
80	TCP	Server health checks.
123	UDP	NTP synchronization.
162	UDP	SNMP traps.
443	TCP	Server health checks.
514	UDP	Syslog.
1510	TCP	Simple Health Checks
3403/501	TCP	Failover Heartbeat ports. If SSL is enabled for failover, port 501 is used; otherwise port 3403 is used.HA Heartbeat
3404/502	TCP	Failover Peer-to-Peer command ports. All non-heartbeat failover operations (such as configuration synchronization) use these ports. If SSL is enabled for failover, port 501 is used; otherwise port 3403 is used.HA all non-Heartbeat Peer-to-Peer communications such as configuration synchronization.
5300	UDP	Envoy sites communication.
5301	UDP	Envoy sites communication.

Default Ports Used by Equalizer for Incoming Traffic

Port Number	IP Protocol Number/ Service	Purpose
N/A	ICMP	Ping and traceroute responses.
22	TCP	SSH administrative CLI access.
53	UDP	Envoy DNS service
80	TCP	• HTTP administrative web UI access. • Predefined HTTP service. Only occurs if the service is used by a virtual server.
161	UDP	SNMP queries.
443	TCP	• HTTPS administrative web UI access. Only occurs if the destination address is a network interface’s IP address. • Predefined HTTPS service. Only occurs if the service is used by a virtual server or virtual cluster , and if the destination address is a virtual server or virtual cluster.
3403	TCP	HA Heartbeat (listen on this port)
3404	TCP	HA all non-Heartbeat Peer-to-Peer communications such as configuration synchronization.(listen on this port)
5300	UDP	Envoy sites communication.
5301	UDP	Envoy sites communication.

Appendix E

Networking Translation Between EQ/OS 10.1.x and 10.2.x

Sections within this chapter include:

Networking Translation Between 10.1.x and 10.2.x Systems

Several, significant networking enhancements were made as part of EQ/OS 10.2.x development. These include:

1. Per-subnet static routes have been enhanced to allow the user to specify an optional source IP address.
   - If no source is specified, the static route applies to all traffic originating from the entire subnet.
   - If a source is specified, the static route applies only to traffic originating from the IP address range specified by the source IP parameter (in CIDR format).

     This allows the user to restrict routes to a specific IP address range or a single IP address.
2. The subnet **outbound_nat** parameter has been removed. It is replaced by enhanced Network Address Translation (NAT) capabilities that allow the user to specify an IP address range (or a single IP) and the IP address that will be used as the source IP for outgoing packets on an interface. NAT rules (like static routes) are specified on a per-subnet basis, providing flexibility when configuring routing. Additional information on NAT can be found in "Configuring Outbound NAT" on page 313.
3. The **default_route** (Default Route) subnet parameter used in 10.1.x configurations has been removed.
4. The **def_src_addr** (Default Source Address) flag used in 10.1.x configurations has been removed.
5. Destination networks used in 10.1.x configurations have been removed. Destination networks are now computed automatically by the system according to the static route configuration. No user configuration is needed. In order for destination networks to be properly computed, static routes must be configured as follows:

   If a gateway provides internet connectivity, a static route should be configured with destination 0/0. If a gateway does not provide internet connectivity, a separate static route should be configured for each network reachable via the gateway..

The table below itemizes specific EQ/OS 10.1.x networking configuration scenarios and describes how they translate to the EQ/OS 10.2.x enhancements in an upgrade. In general:

- The **default_route** subnet parameter is translated to a static route with a 0/0 destination (0.0.0.0/0) in a 10.2.x configuration.
- The **outbound_nat** subnet parameter is translated to a subnet NAT rule which NATs the entire subnet range out the **outbound_nat** IP address.
- When a misconfiguration is detected that results in the new configuration possibly working differently than the old configuration, an error is logged to **/var/log/eq**, as well as the system console. This error will appear after rebooting the system after the upgrade.

EQ/OS 10.1.x Configuration	EQ/OS 10.2.x Configuration	What happens in an upgrade to 10.2.x
Network 1: def_src_addr Default route →GW	Network 1: Static route: 0/0 → GW	In the 10.1.x configuration the combination of the **def_src_addr** flag and a default route meant that *the network that was connected to the Internet*. In 10.2.x configuration, a single static 0/0 route is used.
Network 1: Default route → GW	Network 1: Static route: 0/0 → GW	This is a misconfiguration in 10.1.x configuration since the system is not configured to be connected with *the outside world*. However, the intent of the user is obvious, so it is converted; the same as the case above. An error is not logged.
Network 1: def_src_addr Default route → GW1 Network 2: Default route → GW2	Network 1: Static route: 0/0 → GW1 Network 2: No static routes created at upgrade time	This is a misconfiguration in 10.1.x configuration because the user has not specified all of the connected networks for Network 2. If there is a default route, there should have been a list of destination networks. Since there is not, there is no way of knowing how to convert the Network 2 configuration. An error is logged.
Network 1: Default route → GW1 Network 2: Default route → GW2	Network 1: No static routes created at upgrade time Network 2: No static routes created at upgrade time	This is a misconfiguration in 10.1.x configuration because the user did not specify the connected networks for either Network 1 or Network 2. There is neither a **def_src_addr** flag or a list of destination networks. There is no way of knowing which network is connected to the Internet or any other network. An error is logged.
Network 1: Default route → GW1 Destination network: 0/0 Network 2: Default route → GW2 Destination network: 192/24 Network 3: Static route :172/16 → GW3	Network 1: Static route: 0/0 → GW1 Network 2: Static route: 192/24 →GW2 Network 3: Static route: 172/16 → GW3	This is the proper configuration for a multi-network configuration with multiple, routed networks. Each network has all of its destination networks fully defined, and hence, can converted them properly in an upgrade to 10.2.x.

EQ/OS 10.1.x Configuration	EQ/OS 10.2.x Configuration	What happens in an upgrade to 10.2.x
Network 1: Default route → GW1 Destination network: 0/0 Network 2: Destination network: 192/24	Network 1: Static route: 0/0 → GW1 Network 2: No static routes created at upgrade time	This is a misconfiguration in the 10.1.x configuration because a route has not been specified for communicating with the destination network, 192/24. Because of this, in an upgrade, there is no way of knowing which gateway to use for the 192/24 route. An error is logged.
Network 1: Default route → GW1 def_src_addr Network 2: No destinations or routes	Network 1: Static route: 0/0 → GW1 Network 2: No static routes	Any network with no destinations or routes configured continues to function the same way in 10.2.x configuration. Only Network 1 is connected to outside networks, so it is the only one converted.
Network 1: Default route → GW1 Destination network: 0/0 Network 2: Static route: 192/24 → GW2 Network 3: Static route: 172/16 → GW3	Network 1: Static route: 0/0 → GW1 Network 2: Static route: 192/24 → GW2 Network 3: Static route: 172/16 → GW3	This is a misconfiguration in 10.1.x configuration, since there are no configured destination networks available from Network 2 and Network 3. However, the intent of the user is obvious, so the networking is converted correctly in an upgrade An error is not logged.
Network 1: Static route: 192/24 Destination network: 0/0	Network 1: Static route: 192/24	This is a misconfiguration in 10.1.x configuration, since the destination network is wider than the routing that is available. However, since the intent is obvious, it is converted correctly in an upgrade Only systems reachable through the static route will be reachable, however, those were the only ones that would be reachable in a 10.1.x configuration. An error is not logged.
Network 1: Static route: 192/24 Destination network: 172/16	Network 1: Static route: 192/24	This is a misconfiguration in 10.1.x configuration since the destination network does not match the routing that is available. The resulting configuration is the same as ignoring the destination network. Only systems reachable through the static route will be accessed. An error is logged.

Appendix F
HTTP and HTTPS Header Editing Feature

Sections within this chapter include:

Header Editing

Header editing allows you to add, modify, and delete Layer 7 packet header data contained in client requests and server responses. You can choose to apply header editing rules on every request or response, or you can selectively apply header edits based on whether or not the client request is selected by a match rule.

Header editing is supported on Layer 7 HTTP and HTTPS clusters only.

Edits are defined using a server side scripting language, similar to PHP, that allows you to create custom scripts with a set of rich locator and editing functions that let you easily select headers, locate and modify specific header data, and use that data to add or modify additional headers. Among the operations you can perform are:

- Mask server information such as server version.
- Update request URIs to accommodate path changes on servers. For example, you could change paths from /marketing to /departments/marketing.
- Work around broken features on the server. For example if compression were broken on a server, you could delete gzip from the accept-encoding header.
- Make changes to a query string. For example, you may wish to extract a session ID from a cookie and add it to the query string before sending a request on to a server.

About this Chapter

This document features:

- A Functional Description of the header editing function.
- Definitions of the custom header editing functions and variables and descriptions of how they should be used.
- A description of the scripting language syntax and its use.
- Descriptions of using the scripting language to locate, insert, edit, replace, and delete HTTP/HTTPS header fields.
- Instructions for using Equalizer's CLI and GUI to generate header editing rules.
- Examples showing typical uses.

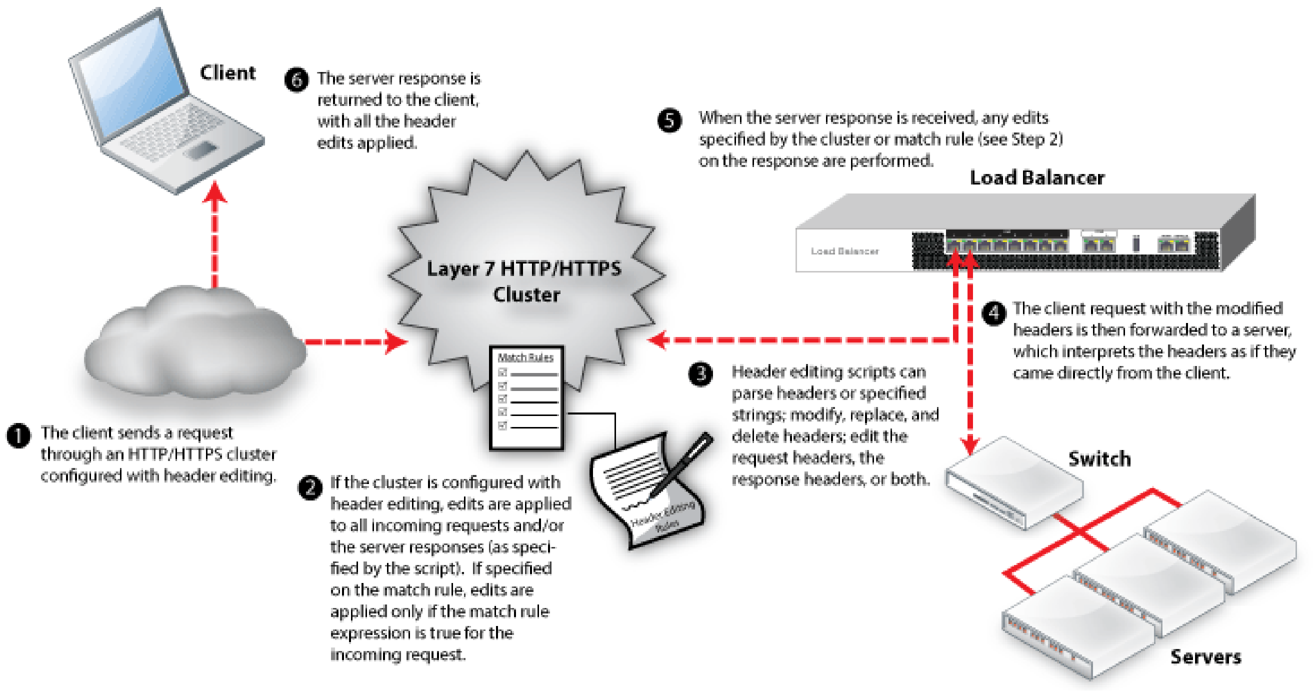
Functional Description

Header Editing can be configured at either the cluster or match rule level:

- When a cluster is configured with header editing rules, *all* client requests and server responses are subject to the rules.
- When a match rule is configured with header editing rules, the header edits are applied only if the match rule expression evaluates to true for an incoming request.

Header editing rules can be applied to client requests, server responses, or both.

The following illustration shows the header editing process from client request to server response.


Client
6 The server response is returned to the client, with all the header edits applied.
5 When the server response is received, any edits specified by the cluster or match rule (see Step 2) on the response are performed.
Load Balancer
Layer 7 HTTP/HTTPS Cluster
Match Rules
4 The client request with the modified headers is then forwarded to a server, which interprets the headers as if they came directly from the client.
3 Header editing scripts can parse headers or specified strings; modify, replace, and delete headers; edit the request headers, the response headers, or both.
1 The client sends a request through an HTTP/HTTPS cluster configured with header editing.
2 If the cluster is configured with header editing, edits are applied to all incoming requests and/or the server responses (as specified by the script). If specified on the match rule, edits are applied only if the match rule expression is true for the incoming request.
Header Editing Rules
Switch
Servers

Header editing can be configured using either the CLI or the GUI. A set of locator and editing functions are provided that let you find specific text in the headers, assign found text to a variable, and use that text to modify other headers in the client request or the server response. The following are some examples of the kinds of edits possible:

- Changing a directory name, file extension, or the complete path in the URI.
- Stripping out fields and headers.
- Adding custom headers.
- Modifying and inserting cookies.
- Inserting text from the request headers in the response headers.

First, we'll explain the features and functions available in the header editing language, then we'll tour the CLI and GUI interfaces.

Header Editing Basics

When you create header edits, you place the search and edit functions within one of two meta-functions:

client_request_edit

All of the searches and edits specified here are made against the headers in the client request, and are performed before the client request is sent to a server behind the ADC. Values saved in variables will be available for use in the **server_response_edit** meta-function as well.

server_response_edit

All of the searches and edits specified here are made against the headers in the server response, and are performed on the headers in the server response after it is received by the ADC. Variables assigned values during client request editing can be used to modify the response header.

These meta-functions can be defined on a cluster, or on one or more match rules. Header edits specified on the cluster are performed on all client requests and server responses. Header edits specified on a match rule are performed only if the incoming request matches the match rule expression.

The following is an example of a **client_request_edit** and **server_response_edit**. Refer to "Header Editing Examples" on page 977 for a detailed description of these edits.

```
client_request_edit()
{
        $xf = header_value("X-Forwarded-For");
        $xf_save = header_value("X-Forwarded-For");
        str_insert($xf, "," . $CLIENT_IP, AFTER);
}
server_response_edit()
{
        header_insert("X-proxy-IP", $xf_save);
}
```

d

Header Editing Function Definitions

The following are definitions for the set of functions used for locating and editing HTTP client request and server response headers. They are used together with user and system variables to construct header editing rules.

Refer to "User-Defined Variables" on page 955 and "System Variables" on page 956 for descriptions of the variables.

header_value(name)

The **header_value( )** function locates a header line by its **name.**

The **name** argument is a string that identifies the name of a header to locate. If the named header is found, the function returns the location of the header value string, which is then assigned to a user-defined variable (See "User-Defined Variables" on page 955).

A header line is expected to be in the following format:

Header-Name: Header-Value

The name argument must match the **Header-Name** without the colon, and the returned location refers to the **Header-Value** text without the "\r\n" (carriage return and newline) characters at the end. If the named header is not found, the location is NULL and its value is an empty string.

header_insert(name, value, *locator*, *whence*)

The **header_insert( )** function inserts a header line into a client request or server response. The **name** argument specifies the header name, and the **value** argument specifies its value.

The **value** argument can be:

- a string literal
- a text replacement variable
- a combination of string literals and variables that are separated by '.' (dot) operators. (See "Using Variables in Replacement Strings" on page 959.)

The optional ***locator***and ***whence*** arguments specify where to insert the header.

The ***whence*** argument must be either of two keywords; BEFORE or AFTER, which indicate whether to add the header before or after the header whose location is indicated by the ***locator*** argument. The ***locator*** is a user defined variable that refers to the location returned from a previous call to a search function. If the ***whence*** argument is provided without a ***locator*** argument, then the keywords BEFORE or AFTER indicate whether the new header is to be inserted before the first or after the last header in the client request or server response. If both optional arguments are omitted, the new header is inserted after the last header.

If the **locator** argument is provided, and it refers to a NULL location because a previous search failed, then the header is not inserted, as it cannot be determined where to apply the edit.

Note - Some functions have multiple optional arguments, which are always specified last. You can omit optional arguments by calling the function without them. However, you cannot skip an optional argument and provide another optional argument that comes afterward. For example, you cannot specify a "locator" argument to the header_insert() function (shown below) without also specifying a "whence" argument because the argument ordering would be incorrect, resulting in a syntax error.

header_replace(name, value)

The **header_replace( )** fu`nction replaces a header line within a client request or server response.

The **name** argument specifies the name of the header to be replaced, and the **value** argument specifies its new value. The **value** argument can be:

- a string literal
- a text replacement variable
- a combination of string literals and variables that are separated by '.' (dot) operators. (See "Using Variables in Replacement Strings" on page 959.)

This function first locates the header line, and then it replaces its value with the new value. This function has no effect if the header to be replaced is not found.

header_delete(name)

The **header_delete( )** function deletes a header line from a client request or server response.

The **name** argument specifies the name of the header to be deleted. The function locates the header line and deletes it. It silently fails if the header to be deleted is not found.

str_find_prefix(locator, string)

The **str_find_prefix( )** function searches for the substring **string** at the beginning of the specified location.

The **locator** argument identifies the text location to search. It can be a user-defined variable that refers to an entire header value or a subset of it. It can also be a system-defined variable that refers to a URI component or a user-defined variable that refers to a subset of a URI component.

The **string** argument specifies the text to search for at the beginning of the desired location.

The function returns the location of the text that was found, which is then assigned to a user-defined variable. This location refers to the entire length of the found text, not just its starting point.

This function is mainly used for marking a string prefix for editing. If the text is not found, this function returns a NULL locator whose value is an empty string.

str_find_suffix(locator, string)

The **str_find_suffix( )** function searches for the substring **string** at the end of the specified location. (e.g., .jpg, .gif, or .txt)

The **locator** argument identifies the text location to search. It can be:

- A user-defined variable that refers to an entire header value or a subset of it.
- One of the system-defined URI variables described in "System Variables" on page 956.

The **string** argument specifies the text to search for at the end of the desired location.

If the location specified by the **locator** argument is found, the function returns the location of the text that was found, which is then assigned to a user-defined variable. This location refers to the entire length of the found text, not just its starting point.

If the text is not found, this function returns a NULL locator whose value is an empty string.

str_find_substr(locator, string)

The **str_find_substr( )** function searches for the substring **string** anywhere within a specified location.

The **locator** argument identifies the text location to search. It can be:

- A user-defined variable that refers to an entire header value or a subset of it.
- One of the system-defined URI variables described in "System Variables" on page 956.

The **string** argument specifies the text to search for anywhere within the desired location.

If the location specified by the **locator** argument is found, the function returns the location of the text that was found, which is then assigned to a user-defined variable. This location refers to the entire length of the found text, not just its starting point.

If the text is not found, this function returns a NULL locator whose value is an empty string.

str_find_regex(locator, string)

The **str_find_regex( )** function searches for a substring that matches the regular expression that is described by **string** within the specified location.

The **locator** argument identifies the text location to search. It can be:

- A user-defined variable that refers to an entire header value or a subset of it.
- One of the system-defined URI variables described in "System Variables" on page 956.

The **string** argument specifies a regular expression string that describes the text to search for at the desired location.

If the **string** is found within the **locator** text, this function returns the location of the string, which is usually assigned to a user variable.

If the **string** is not found in the **locator** text, this function returns a NULL **locator** whose value is an empty string.

str_find_item(locator, string, *delimiter*)

The **str_find_item( )** function searches for a string of characters within the specified **locator**.

The **locator** argument identifies the text location to search. It can be:

- A user-defined variable that refers to an entire header value or a subset of it.
- One of the system-defined URI variables described in "System Variables" on page 956.

The **string** argument specifies an expression that describes the text to search for at the desired location.

The ***delimiter*** argument is an optional argument that specifies the character separating the items in the list. The list may contain white space characters around the delimiter. If the ***delimiter*** is not specified, the default delimiters are used: comma (,), semicolon (;) or ampersand (&).

If the **string** is found within the **locator** text, this function returns the location of the string, which is usually assigned to a user variable.

If the **string** is not found in the **locator** text, this function returns a NULL **locator** whose value is an empty string.

str_find_item_by_name(locator, string, *delimiter*)

The **str_find_item_by_name( )** function searches for a string of characters within the specified **locator.**

The **locator** argument identifies the text location to search. It can be:

- A user-defined variable that refers to an entire header value or a subset of it.
- One of the system-defined URI variables described in "System Variables" on page 956.

The **string** argument specifies an expression that describes the text to search for in the **locator** text

The ***delimiter*** argument is an optional argument that specifies the character separating the items in the list. The list may contain white space characters around the delimiter. If the ***delimiter*** is not specified, the default delimiters are used: comma (,), semicolon (;) or ampersand (&).

If the **string** is found within the **locator** text, this function returns the location of the string, which is usually assigned to a user variable.

If the **string** is not found in the **locator** text, this function returns a NULL **locator** whose value is an empty string.

str_find_item_value(locator, string, *delimiter*)

The **str_find_item_value( )** function searches for a string of characters within the specified **locator**.

The **locator** argument identifies the text location to search. It can be:

- A user-defined variable that refers to an entire header value or a subset of it.
- One of the system-defined URI variables described in "System Variables" on page 956.

The **string** argument specifies an expression that describes the text to search for in the **locator** text

The ***delimiter*** argument is an optional argument that specifies the character separating the items in the list. The list may contain white space characters around the delimiter. If the ***delimiter*** is not specified, the default delimiters are used: comma (,), semicolon (;) or ampersand (&).

If the **string** is found within the **locator** text, this function returns the location of the string, which is usually assigned to a user variable.

If the **string** is not found in the **locator** text, this function returns a NULL locator whose value is an empty string.

str_insert(locator, value, *whence*)

The **str_insert( )** function inserts text within a header line or URI component.

The **locator** argument identifies the text location to search. It can be:

- A user-defined variable that refers to an entire header value or a subset of it.
- One of the system-defined URI variables described in "System Variables" on page 956.

The **value** argument specifies the text to insert which can be a string literal, a text replacement variable, or a combination of string literals and variables separated by '.' (dot) operators (see "Operators" on page 957).

The **whence** argument uses either the BEFORE or AFTER keywords. This indicates whether to insert the text BEFORE or AFTER the text location identified by the locator argument. The **whence** argument is optional and defaults to AFTER if it is not provided.

If the location specified by the **locator** argument is found, the text from the **value** argument is inserted.

If the **locator** argument refers to a NULL location, then no text is inserted, as it cannot be determined where to apply the edit.

str_replace(locator, value)

The **str_replace( )** function replaces text within a header line or URI component.

The **locator** argument identifies the text location to search. It can be:

- A user-defined variable that refers to an entire header value or a subset of it.
- One of the system-defined URI variables described in "System Variables" on page 956.

The **value** argument specifies the replacement text. This can be a string literal, a text replacement variable, or a combination of string literals and variables separated by '.' (dot) operators (see "Operators" on page 957).

If the location specified by the **locator** argument is found, the text from the **value** argument replaces text in the header line.

If the **locator** argument refers to a NULL location, then no text is replaced as it cannot be determined where to apply the edit.

str_delete(locator)

The **str_delete( )** function deletes text from a specified locator.

The **locator** argument identifies the text location to search. It can be:

- A user-defined variable that refers to an entire header value or a subset of it.
- One of the system-defined URI variables described in "System Variables" on page 956.

If the location specified by the **locator** argument is found, the text found in the location is deleted.

If the **locator** argument refers to a NULL location, then no text is deleted as it cannot be determined where to apply the edit.

str_insert_item(locator, item, delimiter, *whence*)

The **str_insert_item( )** function inserts an item into a list of text items separated by delimiters within a header line or URI component.

The **locator** argument identifies the text location to search. It can be:

- A user-defined variable that refers to an entire header value or a subset of it.
- One of the system-defined URI variables described in "System Variables" on page 956.

The **item** argument specifies the text item to insert. It can be a string literal, a text replacement variable, or a combination of string literals and variables separated by "." (dot) operators.

Unlike most other header editing functions, the **delimiter** argument is mandatory and specifies the delimiter character that separates the items in the list. It must be: comma (,), semicolon (;) or ampersand (&).

The ***whence*** argument is optional. It can be either the BEFORE or AFTER keywords.

- If the **locator** argument refers to an individual item, then the ***whence*** argument indicates whether to insert the new item BEFORE or AFTER the located item.
- If the **locator** argument refers to an entire list, then the ***whence*** argument indicates whether the new item should become the first or last item in the list.
- If the ***whence*** argument is omitted, it defaults to AFTER. The function inserts the new item into the list and adds a **delimiter** in the correct place, unless the new item is the only item in the list. White space may be added along with the **delimiter** to make the new item consistent with the rest of the list.

If the location specified by the **locator** argument is found,the text from the **item** argument is inserted.

If the **locator** argument refers to a NULL location, then no text item is inserted, as it cannot be determined where to apply the edit.

str_delete_item(locator)

The **str_delete_item( )** function deletes text items within a header line or URI component.

The **locator** argument specifies a user-defined variable that identifies a text location containing the item that is to be deleted from a list.

If the location specified by the **locator** argument is found, this function deletes the located text.

If the **locator** argument refers to a NULL location, then no text is deleted, as it cannot be determined where to apply the edit.

str_replace_item(locator, item)

The **str_replace_item( )** function replaces an item in a list of text items separated by delimiters within a header line or URI component.

The **locator** argument specifies a user-defined variable that identifies a location containing the item that is to be replaced.

The **item** argument specifies the new item and can be:

- a string literal
- a text replacement variable
- a combination of string literals and variables separated by "." (dot) operators (see "Operators" on page 957).

If the location specified by the **locator** argument is found, this function replaces the previous item in the list with the new item.

If the **locator** argument refers to a NULL location, then the **item** is not replaced, as it cannot be determined where to apply the edit.

str_replace_item_value(locator, item_value)

The **str_replace_item_value( )** function replaces the value part of a name=value pair in a list of text name=value pairs separated by delimiters within a header line or URI component.

The **locator** argument specifies a user-defined variable that identifies a text location containing the item that is to be replaced.

The **item_value** argument specifies the new value. It can be a string literal, a text replacement variable, or a combination of string literals and variables separated by "." (dot) operators. (see "Operators" on page 957).

If the location specified by the **locator** argument is found, this function replaces the previous value part of the name=value pair with the new value, leaving the adjacent equal sign (=) .

If the **locator** argument refers to a NULL location, or to a location that does not contain an equal sign (=) immediately before it, then the text value is not replaced, as it cannot be determined where to apply the edit.

Header Editing Variables

Header editing variables are used together with the editing functions described previously in "Header Editing Function Definitions" on page 945.

There are two types of header editing variables supported: user-defined and system variables. These are explained in the following two sections.

User-Defined Variables

A user-defined variable is used to hold the result of a header editing search function that returns a value, such as:

- a text location that has a starting point and a length
- an entire header value or URI component
- a subset of a header value or URI component

User-defined variables have names that start with a dollar sign ($) followed by 1 to 15 characters. The first character must be a lower-case letter, and the remaining characters can be lower-case letters, numbers or underscores.

User-defined variables can be used as a locator argument to a search or edit function, and to specify text to be used in replacement strings.

Variable assignments made in the **client_request_edit()** meta-function persist until the server-side function is executed, and can be used in the **server_response_edit()** meta-function in text replacement strings.

There is no limit on the number of user-defined variables that can be created in a header edit script.

A user-defined variable cannot be re-defined. For example:

```
$x = header_value("Connection");
$x = header_value("Cookie");
```

This is not allowed and will be identified as an error.

A user-defined variable cannot be used before a value has been assigned to it. It must appear on the left side of an equal sign—before it can be used as an argument to a function. A compiler error will occur and will be displayed if you use a user-defined variable before a value is assigned to it.

System Variables

System variables are used together with the functions and user defined variables to generate header editing rules.

System variables cannot be re-defined. In other words, they cannot be used on the left side of an assignment operator in a script.

There is a set of system-defined variables that are provided. **$DIRNAME**, **$FILENAME**, **$PATHNAME**, and **$QUERY** refer to the URI components of a client request. These variable are provided for the sake of convenience, in order to avoid the need to search for them. These URI variables can be used for editing the URI components, and they can be copied elsewhere by referencing them in text replacement strings.

The other variables provide details about the client, server and cluster. They cannot be edited, but they can be used in text replacement strings.

The following variables are available:

$CLIENT_IP	Client IP address
$CLIENT_PORT	Client port number
$CLUSTER_IP	Cluster IP address
CLUSTER_PORT	Cluster port number
$DIRNAME	URI directory name
$FILENAME	URI file name
$PATHNAME	URI full path name
$QUERY	URI query string. "request=html" causes the '?' delimiter to be added between the pathname and query components of the URI.
$SERVER_IP	Server IP address
$SERVER_PORT	Server port number
$SPNAME	Server pool name

To demonstrate the identification of a URI component, suppose a client sends the following request:

```
GET /data/files/index.html?user=steve&item=pencil HTTP/1.1
Host: 1.1.1.1
Connection: close
```

In this example, the variable assignments will be as follows:

```
$PATHNAME = "/data/files/index.html"
$DIRNAME  = "/data/files/"
$FILENAME = "index.html"
$QUERY    = "user=steve&item=pencil"
```

Each of these variables refers both to the URI component's location and its value. If a URI component is not specified in the client request, it still has a known location and a value of empty string (“”). Because we know where these URI components are to be located, a header editing rule to add a missing URI component. Suppose the client sends the following request:. Suppose the client sends the following request:

```
GET / HTTP/1.1
Host: 1.1.1.1
Connection: close
```

For this example, the variable assignments will be as follows:

$PATHNAME = "/"

$DIRNAME = "/"

$FILENAME = ""

$QUERY = ""

Each of these variables is always defined, so that a value can be inserted into the client request URI by the editing functions.

On the other hand, a user-defined variable may or may not have a location. A user-defined variable has a location if a search matches the text being searched for, but a user-defined variable has no location (location = NULL) if a search could not identify the text being searched for.

Operators

The operators used in Header Editing are:

= (equals)	used for variable assignment for function returns.
. (dot)	concatenation operator for use between two string parameters, either or both of which could be a variable.

Use of White space

White spaces around the '.' operator are optional.

Expression Syntax

There are two types of expressions supported by header editing:

1. Locator expressions
2. Editing expressions

It is important to follow the syntax used with Locator and Editing expressions or errors will occur. You will be prompted to fix the rule syntax if it is determined to be invalid.

Locator Expressions

A locator expression is composed of a variable name, followed by an equal sign, followed by a locator function call, and terminated by a semicolon.

```
$variable = locator_function(arguments);
header_insert("header name", AFTER, $variable);
```

A user-defined variable cannot be used before a value has been assigned to it. For example, the variable `$variable` must be assigned a value—it must appear on the left side of an equal sign—before it can be used as an argument to the `header_insert()` function above.

Editing Expressions

An editing expression is composed of an editing function call followed by a semicolon.

```
editing_function(arguments);
```

For example:

```
str_replace($sound, "howl");
```

Replacement strings can contain string literals and variables, separated by "."s (dots) as explained in the following section.

Using Variables in Replacement Strings

Header editing variables used in rules refer to text locations. To demonstrate this, suppose that we want to use a system-defined variable as replacement text:

```
str_replace($FILENAME, "newfile");
str_replace($file, $FILENAME);
```

In this example, the first line changes the value of `$FILENAME` to "`newfile`", and the second line replaces the contents of $file with the original value of `$FILENAME`.

Locating, Inserting, Replacing, and Deleting Headers

The following sections describe how to locate headers and URI components, inserting, replacing and deleting entire headers, searching within a headers or URI components, and editing within headers or URI components.

Refer to "Header Editing Function Definitions" on page 945 definitions of the functions used in this section.

Note - It should be noted that searches are case-insensitive . All of the text search functions described in this chapter, including the `header_value()` function, are case-insensitive. There is no way to toggle between case sensitive and case insensitive and no way to specify that a single search should be case sensitive. In other words, the header editing language does not provide the `ignore_case()` and `observe_case()` functions that are provided for match rules.

Locating Headers

The `header_value(name)` function is used to locate a header line. It uses a name argument to identify the name of the header to locate. If the named header is found, the function returns the location of the header value string, which is then assigned to a user-defined variable.

To demonstrate how this works, suppose a server returns the following response to a client request:

```
HTTP/1.1 200 OK
Date: Tue, 28 Oct 2014 16:07:07 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Wed, 10 Jul 2013 23:26:04 GMT
ETag: "dbf7-8-4e13099ef63ff"
Accept-Ranges: bytes
Content-Length: 8
Vary: Accept-Encoding
Connection: close
Content-Type: text/html
```

Use the following to locate the value of the `Connection` header:

```
$connection = header_value("Connection");
```

The variable to which the return value of the locator function is assigned (in the example above, `$connection`) refers to two things:

- the location of the header value
- the text contents at that location

In the example above, the location spans from the beginning of header to the end ("Connection: close"). Other functions might return only part of the header data.

If there is no header with the supplied name found, then the return variable is assigned a location of NULL. If more than one header with the supplied name exists, only the first header with that name is parsed; subsequent occurrences are ignored.

Although HTTP and HTTPS permit a header to span multiple lines, this is not a common practice. The header editing rules herein will not act upon more than one line. If a header does span more than one line, the `header_value()` all locator functions will ignore any text that comes after the first line.

Inserting, Replacing and Deleting Entire Headers

The following functions allow you to insert a new header, replace a header, or delete a header entirely:

```
header_insert(name, value, whence, locator)
header_replace(name, value)
header_delete(name)
```

The `header_insert()` function inserts a header into a client request or server response.

The `name` argument specifies the header name, and the `value` argument specifies its new value.

The optional *`whence`* and *`locator`* arguments specify where to insert the header. The `whence` argument must be either of two keywords (BEFORE or AFTER) which indicate whether to add the header *before* or *after* the header whose location is indicated by the *`locator`* argument. If the *`whence`* argument is provided without a `locator` argument, then the keywords BEFORE or AFTER indicate whether the new header is to be inserted before the first or after the last header in the client request or server response. If both optional arguments are omitted, the new header is inserted after the last header.

The optional *`locator`* argument is a user-defined variable that refers to text that was located by the `header_value()` function or one of the text search functions that follow. The *`locator`* argument, if provided, must refer to an existing header, and it cannot be specified without also specifying whether the new header is to be inserted before or after it.

Function	Description
header_insert()	The `header_insert()` function can find the correct location to insert the header even if the locator refers to the middle of a header line. However, if the `locator` is NULL, because a search failed, then the `header_insert()` function has no effect, as it cannot determine where to apply the edit.
header_replace()	The `header_replace()` function replaces a header line within a client request or server response. The `name` argument specifies the header name, and the `value` argument specifies its new value. This function first locates the header line and then it replaces its `value` with the new value. This function has no effect if the header to be replaced is not found.
header_delete()	The `header_delete()` function deletes a header line from a client request or server response. The `name` argument specifies the header name. This function locates the header line and deletes it. This function has no effect if the header to be deleted is not found.

Here's an example of using each function; suppose a server sends back the following response to a client request:

```
HTTP/1.1 200 OK
Date: Tue, 28 Oct 2014 16:07:07 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Wed, 10 Jul 2013 23:26:04 GMT
ETag: "dbf7-8-4e13099ef63ff"
Accept-Ranges: bytes
Content-Length: 8
Vary: Accept-Encoding
Connection: close
Content-Type: text/html
```

Use the following to insert a header after the "`ETag`" header:

```
$etag = header_value("ETag");
header_insert("Cookie", "cat=meow; duck=quack; frog=ribbit", AFTER,
$etag);
```

The value of the `Connection` header is replaced with "`Keep-Alive`":

```
header_replace("Connection", "Keep-Alive");
```

Use the following to delete the "`Date`" header:

```
header_delete("Date");
```

After editing, the following will result:

```
HTTP/1.1 200 OK
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Wed, 10 Jul 2013 23:26:04 GMT
ETag: "dbf7-8-4e13099ef63ff"
Cookie: cat=meow; duck=quack; frog=ribbit
Accept-Ranges: bytes
Content-Length: 8
Vary: Accept-Encoding
Connection: Keep-Alive
Content-Type: text/html
```

You can rename a header(change its name as opposed to its value) by deleting the header and creating a header with a new name and the original value. For example, you can rename the “Server” header to “Waiter” as follows:

```
$server = header_value("Server");
header_insert("Waiter", $server, BEFORE, $server);
header_delete("Server");
```

This removes the “`Server`” header shown above and adds the following in its place:

```
Waiter: Apache/2.2.22 (Ubuntu)
```

This example also shows how `$server` refers to both a location and its contents. In the call to `header_insert()` above, specifying `$server` as the value argument refers to the text string "Apache/2.2.22 (Ubuntu)", and specifying `$server` as the `locator` argument refers to its location. Whether used as a locator or a value, a user-defined variable always refers to the original header, as it was received from the client or server, and not its value after editing rules have been applied. For this reason, the `header_insert()` and `header_delete()` calls above could be reversed, and the result would be the same:

```
$server = header_value("Server");
header_delete("Server");
header_insert("Waiter", $server, BEFORE, $server);
```

Searching Within a Header or URI Component

The following describes the header editing functions used to search within headers or URI components The `str_find_*()` functions are used to locate text within a header value or URI component string:

```
str_find_prefix(locator, string)
str_find_suffix(locator, string)
str_find_substr(locator, string)
str_find_regex(locator, string)
```

The `locator` argument specifies either a user-defined variable, which refers to all or part of a header value `string`, or one of the system-defined URI variables described in "System Variables" on page 956.

The `string` argument describes the text to search for at a location.

Each of these functions returns the location of the text that was found, which is then assigned to a user-defined variable. This user-defined variable refers to both the location and content of the found text. If the search does not find anything, the user-defined variable has no location (location is NULL) and has a value of empty string ("").

Function	Description
str_find_prefix ()	The `str_find_prefix()` function searches for the substring `string` at the beginning of the specified location.
str_find_suffix()	The `str_find_suffix()` function searches for the substring `string` at the end of the specified location.
str_find_substr()	The `str_find_substr()` function searches for the substring `string` anywhere within the specified location.
str_find_regex()	The `str_find_regex()` function searches for the regular expression that is described by `string` within the specified location.

To demonstrate how these functions work, suppose a client sends the following request:

```
GET /data/files/index.html?user=steve&item=pencil HTTP/1.1
Host: 172.16.246.1
Connection: Keep-Alive
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate, compress
```

Use the following locator functions:

```
$prefix = str_find_prefix($PATHNAME, "/dat");
$suffix = str_find_suffix($FILENAME, ".html");
$encoding = header_value("Accept-Encoding");
$deflate = str_find_substr($encoding, "deflate");
$firstword = str_find_regex($encoding, "^([A-Z][a-z]*), ");
```

In this example, `$prefix` refers to the substring "`/dat`" at the beginning of the pathname component of the URI, `$suffix` refers to the substring "`.html`" at the end of the filename component, `$deflate` refers to the substring `"deflate"` in the middle of the `"Accept-Encoding"` header, and `$firstword` refers to the substring `"gzip"` at the beginning of the `"Accept-Encoding"` header.

Editing Within a Header or URI Component

Once text is located, it can be modified, deleted, or it can be used as a point of reference to add new text. It can also be copied elsewhere. The following string editing functions are used for these tasks:

```
str_insert(locator, value, whence)
str_replace(locator, value)
str_delete(locator)
```

Function	Description
str_insert()	The `str_insert()` function inserts text within a header line or URI component. The "`value`" argument specifies the text to insert, and the `locator` and `whence` arguments specify where to insert the text. If the `locator` is the return value from a call to the `header_value()` function, or a system-defined URI component variable such as `$PATHNAME`, then it refers to the entire header line or URI component. In that case, the `whence` keywords (BEFORE and AFTER) indicate whether to insert the text at the beginning or end of the header line or URI component. Otherwise, if the `locator` is the return value from one of the "`str_find_*()`" functions, then it refers to the substring that was found within the header or URI component by the search. In that case, the `whence` keywords (BEFORE and AFTER) indicate whether to insert before or after the substring. The "*`whence`*" argument is optional and defaults to AFTER if it is not provided. If the `locator` argument is NULL, because a search failed to find something, then the `str_insert()` function has no effect, as it cannot determine where to apply the edit.

Function	Description
str_replace()	The `str_replace()` function replaces text within a header line or URI component. The `locator` argument identifies the location and length of the text to be replaced, and the "`value`" argument specifies the replacement text. If the `locator` is the return value from a call to `header_value()` or a system-defined URI component variable, such as `$PATHNAME`, then it refers to the entire header line or URI component. In that case, the entire header line or URI component is replaced. Otherwise, if the `locator` is the return value from one of the "`str_find_*()`" functions, then it refers to the substring that was found within the header or URI component by the search. In that case, only the substring is replaced. The `str_replace()` function has no effect if the `locator` argument is NULL, as it cannot determine where to apply the edit.
str_delete()	The `str_delete()` function deletes text within a header line or URI component. The `locator` argument identifies the location and length of the text to be deleted. The `locator` can be the return value from a call to the `header_value()` function or a system-defined URI component variable, in which case the edit would produce a header line or URI component with an empty value. More likely, the `locator` is a return value from one of the "`str_find_*()`" functions, and in that case only the specified substring is deleted. The `str_delete()` function has no effect if the `locator` argument is NULL, as it cannot determine where to apply the edit.

To demonstrate this , suppose that a client sends the following request:

```
GET /data/files/document.html HTTP/1.1
Host: 172.16.246.1
Connection: Keep-Alive
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate, compress
```

Use these editing functions to change the filename suffix from ".html" to ".pdf":

```
$suffix = str_find_suffix($FILENAME, ".html");
str_replace($suffix, ".pdf");
```

These lines insert text into the middle of the `Connection` header:

```
$connection = header_value("Connection");
$alive = str_find_substr($connection, "Alive");
str_insert($alive, "Keep-" BEFORE);
```

This replaces the value of the `Accept-Language` header.

```
$lang = header_value("Accept-Language");
str_replace($lang, "ja-jp");
```

These lines edit the `Accept-Encoding` header, changing `deflate` to `flatten`:

```
$encoding = header_value("Accept-Encoding");
$deflate = str_find_substr($encoding, "deflate");
str_replace($deflate, "flatten");
```

You may want to add a query to the URI to remember that the client actually asked for an HTML file:

```
str_replace($QUERY, "request=html");
```

The following is a result of the editing rules:

```
GET /data/files/document.pdf?request=html HTTP/1.1
Host: 172.16.246.1
Connection: Keep-Alive
Accept: */*
Accept-Language: ja-jp
Accept-Encoding: gzip, flatten, compress
```

Note - Changing the value of **$QUERY** from "" to "request=html" caused the '?' delimiter to be added between the pathname and query components of the URI.

For a better example of the `str_find_regex()` function, suppose a server response contains the following header line:

```
Server: Apache/2.2.22 (Ubuntu)
```

The following locates and edits this line:

```
$server = header_value("Server");
$before_slash = str_find_regex($server, "^([A-Z][a-z]*)/");
str_replace($before_slash, "Cherokee");
```

After editing, the header line will appear as follows:

```
Server: Cherokee/2.2.22 (Ubuntu)
```

Using the CLI and GUI to Configure Header Editing Rules

Before using this feature, it is highly recommended that you carefully review "Header Editing Function Definitions" on page 945, "Header Editing Variables" on page 955, **and** "Expression Syntax" on page 958.

Use the CLI or GUI to configure header editing rules. You can add comments to the rules if needed. Refer to "Using Comments with Header Editing Scripts" on page 971 for details.

Configuring Editing Header Rules using the CLI

Configure header editing rules in the CLI as follows:

1. Log in to the CLI.
2. Do one of the following:
   a. To generate header rules and attach them to a cluster, enter the cluster context for the HTTP/HTTPS cluster. When a cluster is configured with header editing rules, *all* client requests and server responses are subject to the rules.
   b. To generate header rules and attach them to a match rule, enter the match rule context for the match rule on which the header editing rules are to be attached. When a match rule is configured with header editing rules, the header edits are applied only if the match rule expression evaluates to true for an incoming request.
3. Use the following syntax to either invoke the editor to generate header editing rules or fetch an existing header editing script from a URL or the file store.
   a. Enter the desired header editing rules and save as a script.(`edit`)
   b. Fetch an existing header editing rule script from a fully qualified FTP or HTTP site. (`url`)
   c. Fetch an existing header from the file store. The path to the header edit script file should be in the format: `file:// script name`

When attaching to a match rule:

```
eqcli cl-clustername*-ma-matchrulename> hdredit <edit|url>
```

When attaching to a cluster:

```
eqcli cl-clustername*> hdredit <edit|url>
```

The following shows the CLI syntax used for either generating header editing scripts or fetching an existing script from an FTP or HTTP site or from the file store. Refer to "Header Editing Examples" on page 977 for practical examples of generating editing rules.

Using the Editor to Create Rules

When you use the `hdredit edit` command, the file editor will be invoked. For example, if you enter `eqcli > eqcli cl-clustername*-ma-matchrulename> hdredit edit` or `eqcli > eqcli cl-clustername* > hdredit edit` the following will be displayed.

```
^[ (escape) menu  ^e search prompt  ^y delete line    ^u up     ^p prev
page
^a ascii code     ^x search         ^z undelete line  ^d down   ^n next
page
^b bottom of text ^g begin of line  ^w delete word    ^l left
^t top of text    ^o end of line    ^v undelete word  ^r right
^c command        ^k delete char    ^f undelete char      ESC-Enter: exit
ee
L: 1 C: 1
=====================================================================
client_request_edit ()
{EDITING RULES
}
server_response_edit()
{EDITING RULES
}
```

The editing rules as described in the previous sections should be entered between the brackets {}. When you are finished, enter **ESC** and follow the prompts to save the script to the file store. When fetching the script, you will need to know the name of the script file.

Refer to "Using the File Editor" on page 927 for additional instructions on using the editor.

Fetching a Header Editing Rule Script using a URL

If you want to fetch an existing header editing rule script from a fully qualified FTP or HTTP location, enter the following to fetch the script and attach it to a cluster or match rule.

```
eqcli cl-clustername*-ma-matchrulename> hdredit ftp://x.x.x.x

or

eqcli cl-clustername*> hdredit ftp://x.x.x.x
```

Fetching a Header Editing Rule Script from the File Store

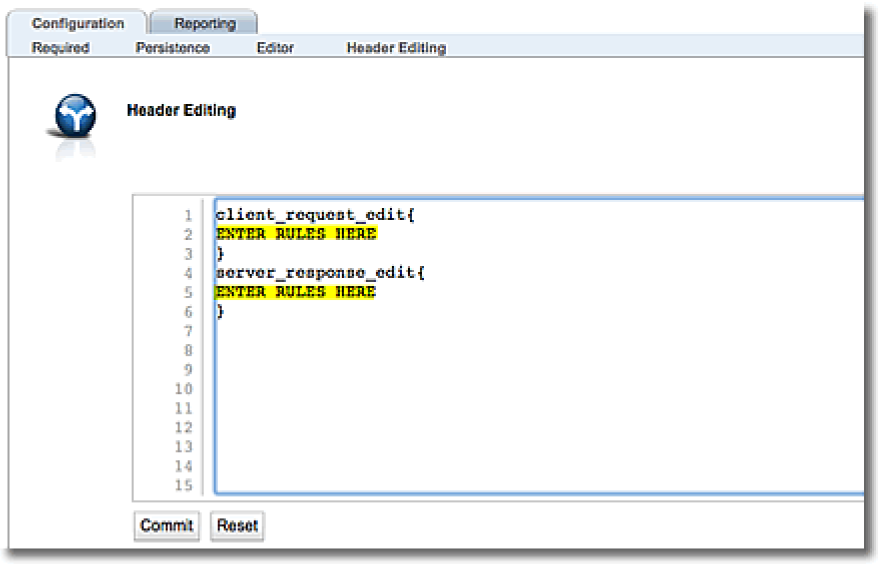
If you used the file editor to save a header editing rule script to the file store, you can fetch the script and attach it to a cluster or match rule as follows.

```
eqcli cl-clustername*-ma-matchrulename> hdredit file://scipt_name

or
-
eqcli cl-clustername*> hdredit file://scipt_name
```

Configuring Editing Header Rules using the GUI

Configure header editing rules in the GUI as follows:

1. Log in to the GUI.
2. Do one of the following:
   a. To edit header rules at a cluster level, select an HTTP/HTTPS cluster on the left navigational pane and then select **Configuration > Header Editing** to display the **Header Editing** rule configuration screen. When displayed, you will see that there are placeholders for **client_requrest_edit** and **server_response_edit**. Header editing rules should be entered between the braces {} for client request and/or server response. When a cluster is configured with header editing rules, *all* client requests and server responses are subject to the rules.
   b. To edit header rules attached to a match rule, select a match rule attached to an HTTP/HTTPS cluster on the left navigational pane and then select **Configuration > Header Editing** to display the **Header Editing** rule configuration screen. The screen is the same as that used for attaching header edits to a cluster. Header editing rules should be entered between the braces {} for **client_request_edit** and/or **server_response_edit**. When a match rule is configured with header editing rules, the header edits are applied only if the match rule expression evaluates to true for an incoming request.



```
client_request_edit{
ENTER RULES HERE
}
server_response_edit{
ENTER RULES HERE
}
```

3. Clicking on the **Reset** button will remove anything that you've entered and return the screen to the default display. Click on **Commit** to save the header editing rules attached to the cluster or match rule.

Refer to "Header Editing Examples" on page 977 for practical examples of generating editing rules.

Using Comments with Header Editing Scripts

When creating header editing scripts in the GUI or CLI, you can add comments to the scripts if necessary. The purpose of using comments is that they can explain the format or intent of a script to anyone reading it and/or make the scripts and their functions easier to understand. A hash character (#) is used as the commenting character and, when used, tells the header edit compiler that the string that immediately follows is a comment and that it should be ignored until the next line that doesn't begin with a #.

You can:

- add comments for entire lines
- add comments that follow search and edit function calls
- comment out script lines

An example of header edit scripts with comments:

Here is an example of using comments with header editing scripts which are used the same with both the GUI or the CLI.

```
# Use this function for client request edits.
#
client_request_edit()
{
        # Search for "X-forwarded-for" header
        $xf = header_value("X-forwarded-for");
        # Add client's IP address to end of "X-forwarded-for" header
        str_insert($xf, "," . $CLIENT_IP, AFTER);
}
# Use this function for server response edits.
#
server_response_edit()
{
        # Search for "Content-Type" header.
        $content = header_value("Content-Type");

        $text = str_find_prefix($content, "text");      # Find "text" prefix
        $html = str_find_suffix($content, "html");      # Find "html" suffix
        str_replace($text, "left");                     # Replace "text" with "left"
        str_replace($html, "right");                    # Replace "html" with "right"
}
```

Commenting Out Script Lines

You can disable a line in a script by "commenting it out" with the # character. Since the # that precedes a line tells the compiler that this is a comment line –the compiler will ignore the line and proceed to the next line. If you enter the following:

```
# header_insert("X-proxy-IP", $xf);
```

The compiler will ignore the header_insert() function call that follows the # character.

Header Editing Reporting

Header Editing statistics are displayed for the cluster or match rule on which they are attached. They are displayed in the Header Edit Statistic group at the bottom of the CLI display.

To display statistics for a cluster enter:

```
eqcli > cluster clustername stats
```

For example:

```
eqcli > cluster cluster_http stats
                Current         60 sec          10 min          60 min
TOTALPRCSD      50891           34              37              27
TOTALRESPPRCSD  68535           60              63              43
TIMESPENT       45526           N/A             N/A             N/A
ACTIVECONX      242             0               256             186
BYTERCVD        20354896        N/A             N/A             N/A
BYTESEND        146733440       N/A             N/A             N/A
DROPNOSRVR      0               N/A             N/A             N/A
TOTALSTKY       0               N/A             N/A             N/A
CURRSTKY        0               0               0               0
REQPARSED       68535           N/A             N/A             N/A
REQFAILED       0               N/A             N/A             N/A
REQFAILHDR      0               N/A             N/A             N/A
RSPPARSED       0               N/A             N/A             N/A
RSPFAILED       0               N/A             N/A             N/A
RSPFAILHDR      0               N/A             N/A             N/A
CLNTTO          68111           N/A             N/A             N/A
SRVRTO          0               N/A             N/A             N/A
CONNTO          0               N/A             N/A             N/A
SELPERSIST      0               N/A             N/A             N/A
SPLICE          50891           N/A             N/A             N/A
CURCLNTWAITQ    0               N/A             N/A             N/A
CURCLNTWAITSRVR 0               N/A             N/A             N/A
CURCOMP         0               0               0               0
TOTALCOMP       0               N/A             N/A             N/A
INBYTECOMP      0               N/A             N/A             N/A
OUTBYTECOMP     0               N/A             N/A             N/A

Header Edit Statistic                                           Count
---------------------------------------------------------------------
Number of times client_request_edit() executed successfully:    2
Number of times client_request_edit() executed with errors:     0
Number of times server_response_edit() executed successfully:   2
Number of times server_response_edit() executed with errors:    0
eqcli >
```

To display statistics fora match rule enter:

```
eqcli > cluster clustername match matchrulename
```

For example:

```
eqcli > cluster cluster_http match test stats
                Current         60sec           10min           60min
TOTALPRCSD      6157678         4218            3028            2479

Header Edit Statistic                                           Count
----------------------------------------------------------------------
Number of times client_request_edit() executed successfully:    2
Number of times client_request_edit() executed with errors:     0
Number of times server_response_edit() executed successfully:   2
Number of times server_response_edit() executed with errors:    0
eqcli >
```

Error Logging

If a search function cannot identify the text being searched for (e.g., value or locator), the user-defined variable that stores the result will be assigned a NULL location. If this variable is passed to an editing function as replacement text, the user defined variable is treated as an empty string and an error message is logged.

For example, if a client request script is as follows:

```
$engineering = str_find_prefix($DIRNAME, "/engineering/");
str_replace($engineering, "/departments/engineering/");
```

The `str_replace()` function will not make any edits if `$engineering` cannot find `"/engineering/"`, as it cannot determine where to apply the edit.

In this case, an error is logged even though the script *can* be executed successfully. However, edits will not be made since the locator is NULL.

The same is true for server response edits. If the search function cannot find the value or locator being searched for in the header from the server response, the result will be NULL and an error will be logged.

The Header Edit statistics accumulate from the time that your ADC has started. If you remove a script or restart your ADC, the statistics counter will be reset to "0".

The following table describes the Header Edit cluster or match rule statistics.

Statistic	Description
`Number of times client_request_edit() executed successfully:`	The number of times that the script has executed successfully and edited client request headers.
`Number of times client_request_edit() executed with errors:`	The number of times that the script has executed successfully, however, logged an error because a locator or value was not found in the client request header.
`Number of times server_response_edit() executed successfully:`	The number of times that the script has executed successfully and edited server response headers.
`Number of times server_response_edit() executed with errors:`	The number of times that the script has executed successfully, however, logged an error because a locator or value was not found in the server response header.

Header Editing Examples

Header editing allows you to add, modify, and delete Layer 7 packet header data contained in client requests and server responses. The following examples demonstrate how to use header editing functions in real-world scenarios.These include:

1. Modifying the directory path portion of a client request URI before sending the request to the servers behind the ADC.
2. Inserting a new header into the list of client request headers.
3. Inserting a client IP into an existing client request header and (in the same script) inserting a new header into the response that comes back from the server (before passing the response back to the client).

Each example includes:

- A clear description of the example's purpose and the desired results.
- How to construct the header edit script used in the example.
- How to configure the header edit script using either the CLI or the GUI.

Example of Updating a URI with an Updated Path

The purpose of this example is to demonstrate how to use header editing to edit the path in a client request header line so that servers receive a different path name. We will change "`/engineering/`" to "`/departments/engineering/`".

In this example, header editing rules for a match rule will be generated. The match rule will evaluate cluster packets to look for the `dirname_prefix("/engineering/")` expression and use the header editing rules attached when the match rule expression evaluates to "True".

The request headers are follows:

```
GET /engineering/files/document.html HTTP/1.1
Host: 172.16.246.1
Connection: Keep-Alive
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, compress
```

You will need to:

1. Create a match rule on a cluster so that the match rule expression selects a directory path of "`/engineering/`".
2. Enter a header editing script for the match rule that will replace the directory path of "`/engineering/`" with "`/departments/engineering/`" for all client request headers. The incoming headers received by servers will be:

   ```
   GET /departments/engineering/files/document.html HTTP/1.1
   Host: 172.16.246.1
   Connection: Keep-Alive
   Accept: */*
   Accept-Language: en-us
   Accept-Encoding: gzip, compress
   ```

Script Construction

For this example, we'll construct a script that will locate the directory name we're looking for and modify it as outlined in the previous section. We'll use the following functions and variables in the script:

- We'll use the `string_find_prefix()` function (See "str_find_prefix(locator, string)" on page 947) to locate the '/engineering/' directory in the request URI.
- We'll add the additional directory to the URI using the `str_replace()` function (See "str_ replace(locator, value)" on page 951).
- The `$DIRNAME` (Directory Name) system variable is used to identify a directory path.

The following client request script is used to edit the incoming client request headers:

```
$engineering = str_find_prefix($DIRNAME, "/engineering/");
str_replace($engineering, "/departments/engineering/");
```

1. The first line of the script defines a user variable (`$engineering`) that uses the `str_find_prefix(`*`locator, string`*`)` function.This function searches for a directory and a specified name for the directory. `$engineering` is the name of the user variable in this example.
   a. The locator argument specifies the header (or part of a header) that we want to search for a particular string. In this case, we use the system variable `$DIRNAME`, which contains the directory portion of the client request URI `string` .
   b. The string argument specifies the text for which we are searching ("`/engineering/`"), which will be replaced as described below.
2. On the second line, the `str_replace(`*`locator, value`*`)` function is used to replace text within a header line:
   a. The `locator` argument specifies the text that we want to replace. In this case, the `$engineering` variable specifies the text and location assigned by the `str_find_prefix()` function.
   b. The `value` argument is the replacement text to use. In this example, "`/departments/engineering/`" is used to replace "`/engineering/`".

The following two sections demonstrate how to attach the header edit script to a match rule using the CLI and the GUI.

Using the CLI

1. Log in to the CLI.
2. Create a match rule that uses a `dirname_prefix("/engineering/")` expression in the definition. In this case, the header edits will be applied only if the incoming client request URI contains a directory prefix of `"/engineering/"`. (See "Creating a New Match Rule" on page 445 for instructions.). You use a configured match rule prior to configuring editing scripts.
3. Enter the match rule context and enter `eqcli` **`cl-`*`clname`*`-ma-`*`maname`* `> hdredit edit`** to activate the file editor.
4. Enter the script described in the previous section between the braces beneath `client_request_edit()`. Note that there are braces beneath `server_response_edit()` as well. You can leave the space between the braces blank or remove the `server_response_edit()` function and braces.

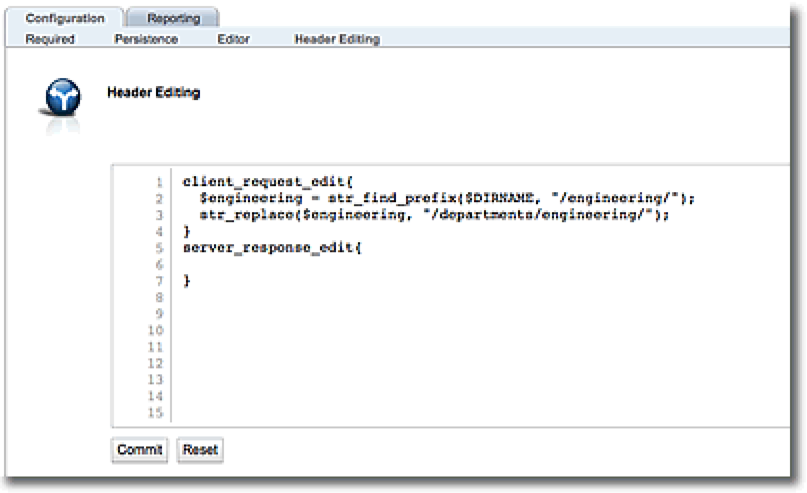
```
eqcli cl-clname-ma-maname>  hdredit edit
^[ (escape) menu  ^e search prompt  ^y delete line    ^u up    ^p prev page
^a ascii code     ^x search         ^z undelete line  ^d down  ^n next page
^b bottom of text ^g begin of line  ^w delete word    ^l left
^t top of text    ^o end of line    ^v undelete word  ^r right
^c command        ^k delete char    ^f undelete char     ESC-Enter: exit ee
L: 1 C: 1
=====================================================================
client_request_edit()
{
        $engineering = str_find_prefix($DIRNAME, "/engineering/");
        str_replace($engineering, "/departments/engineering/");
}
server_response_edit()
{
}
```

5. Enter **`ESC-Enter`** and follow the prompts to exit the file editor and save the header edit.

Using the GUI

As in the CLI, we'll assume that the cluster to which clients will be sending the requests we want to edit already exists. We'll create a new match rule whose expression will search for "`/engineering/`" in an incoming request URI. The header edits will be applied only if this string is found in the URI. (See "str_find_prefix(locator, string)" on page 947).

1. Log in to the GUI.
2. Create a match rule that uses a `dirname_prefix"/engineering/"` expression in the definition. In this case, the header edits will be applied only if the match rule expression identifies a directory prefix of "`/engineering/`" for an incoming request. (See "Creating a New Match Rule" on page 445 for instructions.)
3. Select the match rule on the left navigational pane and then click **Configuration > Header Editing** to display the **Header Editing** configuration screen.
4. Enter the script described in the previous section between the braces beneath `client_request_edit()`. Note that there are braces beneath `server_response_edit()` as well. You can leave the space between the braces blank or remove the `server_response_edit()` function and braces.



```
client_request_edit{
  $engineering = str_find_prefix($DIRNAME, "/engineering/");
  str_replace($engineering, "/departments/engineering/");
}
server_response_edit{

}
```

5. Clicking on the **Reset** button will remove anything that you've entered and return the screen to the default display.
6. Click on **Commit** to save the header editing rules attached to the match rule.

Example of Inserting a New Header into a URI

The purpose of this example is to demonstrate how to use header editing to insert a new header line into every client request received by a cluster.

For this example, we'll assume that the request headers are as follows:

```
GET /data/files/document.html?user=name&sessionid=87654321 HTTP/1.1
Host: 172.16.246.1
Connection: Keep-Alive
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate, compress
```

We'll create a client request header editing script for a cluster that will insert a new "Cookie" header line in the URI. request headers. We want the new header to appear after the "`Accept-Charset`" header, so the headers received by the servers behind the ADC will look like this:

```
GET /data/files/document.html?user=name&sessionid=87654321 HTTP/1.1
Host: 172.16.246.1
Connection: Keep-Alive
Accept: */*
Accept-Language: en-us
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Accept-Encoding: gzip, deflate, compress
```

Script Construction

In this example, client request header edits are used to: locate a specific header, specify what the header line should display when received by servers, and where it should be located.

- The `header_value(name)` function is used to locate a specific header.
- The `header_insert(name, value, *whence*)` function is used to; insert a new header, specify what it should be, and specify the location.

The following client request script is used:

```
        $charset=header_value("Accept-Language");
        header_insert("Accept-Charset", "ISO-8859-1,utf-8;q=0.7,*;q=0.7", AFTER,
$charset);
```

1. In the first expression, the user variable `$charset` is assigned the location and content of the "`Accept-Language`" header in the client request headers.
2. The second expression uses the `header_insert(name, value, whence)` function to insert the new header, as follows:
   a. A `name` argument is specified. In this case the new header line is called "`Accept-Charset`".
   b. A `value` argument is specified. In this case the `value` of the new header line "`Accept-Charset`" is "`ISO-8859-1,utf-8;q=0.7,*;q=0.7`".
   c. A `whence` argument is specified that identifies the location of the text to be replaced. In this case, `AFTER` is used to specify that the header line should be placed AFTER the value identified by the $`charset` variable defined in step 1.

The following two sections demonstrate how to attach the header edit script to a cluster using the CLI and the GUI.

Using the CLI

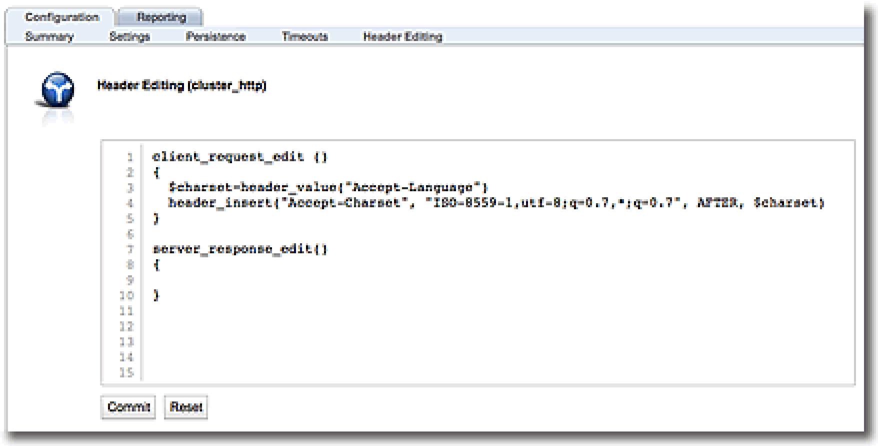
1. Log in to the CLI.
2. Since the header editing scripts will be applied for all client requests through a cluster, the script must be configured on the cluster. Enter the cluster context and enter `eqcli` **`cl-`*`clname`* `> hdredit edit`** to activate the file editor.
3. Enter the script described in the previous section between the braces beneath `client_request_edit()`. Note that there are braces beneath `server_response_edit()` as well. You can leave the space between the braces blank or remove the `server_response_edit()` function and braces.

```
eqcli cl-clname> hdredit edit
^[ (escape) menu  ^e search prompt  ^y delete line    ^u up    ^p prev page
^a ascii code     ^x search         ^z undelete line  ^d down  ^n next page
^b bottom of text ^g begin of line  ^w delete word    ^l left
^t top of text    ^o end of line    ^v undelete word  ^r right
^c command        ^k delete char    ^f undelete char      ESC-Enter: exit
ee
L: 1 C: 1
====================================================================
client_request_edit()
{
   $charset=header_value("Accept-Language");
   header_insert("Accept-Charset", "ISO-8859-1,utf-8;q=0.7,*;q=0.7", AFTER,
$charset);
}
server_response_edit()
{
}
```

4. Enter **`ESC-Enter`** and follow the prompts to exit the file editor and save the header edit.

Using the GUI

1. Log in to the GUI.
2. Select the cluster on the left navigational pane and then select **Configuration > Header Editing** to display the **Header Editing** rule configuration screen.
3. Enter the script described in the previous section between the braces beneath `client_request_edit()`. Note that there are braces beneath `server_response_edit()` as well. You can leave the space between the braces blank or remove the `server_response_edit()` function and braces.



```
client_request_edit ()
{
  $charset=header_value("Accept-Language")
  header_insert("Accept-Charset", "ISO-8559-1,utf-8;q=0.7,*;q=0.7", AFTER, $charset)
}

server_response_edit()
{

}
```

4. Click on the **Test** button to verify that the header edit script compiles successfully. If there is a problem with the script, you will be given the option to re-enter. Clicking on the **Reset** button will remove anything that you've entered and return the screen to the default display.
5. Click on **Commit** to save the script.

Example of Using the X-Forwarded-For Header Field to Insert the Client IP Address into a Client Request.

The `X-Forwarded-For` header field is typically used to identify the originating IP address of a client connecting to a server. The purpose of this example is to demonstrate how to use header editing to make the `X-Forwarded-For` header line insert a client IP address into the client request header for every client request through a cluster. In addition, an `X-proxy-IP` header line will be inserted into the response header. `X-proxy-IP` is a non-standard, user-defined header used for this example. It will display the value of the `X-Forwarded-For` header line from the client request.

In the examples below:

172.16.246.1 is the cluster IP
172.16.6.1 is the ADC IP
10.0.0.11 is the client IP

The request headers are as follows:

```
GET /data/files/document.html HTTP/1.1
Host: 172.16.246.1
Connection: Keep-Alive
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate, compress
X-forwarded-For: 172.16.60.1
```

We want to create a `client_request_edit()` script that will add an `X-Forwarded-For` header line by adding a client IP address that is appended with response IP. We'll also create a `server_response_edit()` script that will be used to insert the response IP and use it for a new `X-proxy-IP` header line.

The client request header will be changed to the following:

```
GET /data/files/document.html HTTP/1.1
Host: 172.16.246.1
Connection: Keep-Alive
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate, compress
X-forwarded-For: 172.16.60.1,10.0.0.11
```

The server response headers will be changed to the following:

```
HTTP/1.1 200 OK
Date: Tue, 28 Oct 2014 16:07:07 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Wed, 10 Jul 2013 23:26:04 GMT
ETag: "dbf7-8-4e13099ef63ff"
Accept-Ranges: bytes
Vary: Accept-Encoding
Connection: close
Content-Type: text/html
X-proxy-IP: 10.0.0.11
```

Script Construction

In this example we'll create `client_request_edit()` expressions that will find the location of the `X-Forwarded-For` header line in the client request.

The `client_response` header edit inserts an `X-proxy-IP` header line at the end of the server response. Its value is the appended value of the `X-Forwarded-For` header in the client request.

The following client request and server response scripts are used:

```
client_request_edit()
{
        $xf = header_value("X-Forwarded-For");
        $xf_save = header_value("X-Forwarded-For");
        str_insert($xf, "," . $CLIENT_IP, AFTER);
}
server_response_edit()
{
        header_insert("X-proxy-IP", $xf_save);
}
```

For the client request header edit:

1. In the first expression, the user variable `$xf` is assigned the location and content of the `X-Forwarded-For` header in the client request headers.
2. In the second expression, the `str_insert(`*`locator`*`, `*`value`*`, `*`whence`*`)` function is used to insert a comma followed by an IP address into the `X-Forwarded-For` header:
    a. The user variable `$xf` (which was assigned a location/value in Step 1) is used as the locator argument. This is the location at which we want the function to insert a string.
    b. For the `value` argument, we specify the string that we want to insert into the header; in this case, a comma followed by an IP address. To specify this in the function call, we use the concatenation operator (".") to tell the compiler to interpret the comma and the value that follows as a single string. Following the concatenation operator is the system variable `$CLIENT_IP`, which always contains the client IP address from the client request headers.
    c. Finally, the `whence` argument specifies where to insert the value into the header, relative to the locator argument (`$xf` in this case). We use the keyword `AFTER` in our function call, to specify that the value string is inserted after the location specified by `$xf`.
3. In the third expression, the user variable `$xf_save` is assigned the location and content of the `X-Forwarded-For` header that will be used in the server response header.

For the server response header:

4. The `header_insert(name, value)` function is used to insert a new header into the server response.

    a. The `name` argument specifies the header name. In this case we want the server response header line to be "`X-proxy-IP`". Since a locator is not specified (is NULL), the header line will be placed at the end of the list of headers in the server response.

    b. The `value` argument specifies its value.

The following two sections demonstrate how to attach a the header edit script described above to a cluster using the CLI and the GUI.

Using the CLI

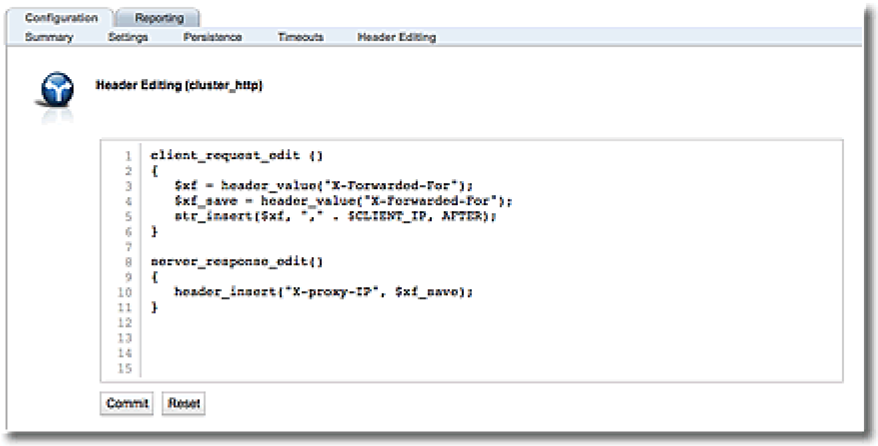
1. Log in to the CLI.
2. Since the header editing scripts will be applied for all client requests through a cluster, the script must be configured on the cluster. Enter the cluster context and then enter `eqcli` **`cl-`*`clname`* `> hdredit edit`** to activate the file editor.
3. Enter the scripts described in the previous section in the new braces beneath `client_request_edit()` and `server_response_edit()` as shown below.

```
eqcli cl-clname > sdhdredit edit
^[ (escape) menu  ^e search prompt  ^y delete line    ^u up    ^p prev page
^a ascii code     ^x search         ^z undelete line  ^d down  ^n next page
^b bottom of text ^g begin of line  ^w delete word    ^l left
^t top of text    ^o end of line    ^v undelete word  ^r right
^c command        ^k delete char    ^f undelete char     ESC-Enter: exit ee
L: 1 C: 1
======================================================================
client_request_edit()
{
        $xf = header_value("X-Forwarded-For");
        $xf_save = header_value("X-Forwarded-For");
        str_insert($xf, "," . $CLIENT_IP, AFTER);
}
server_response_edit()
{
        header_insert("X-proxy-IP", $xf_save);
}
```

4. Type **`ESC-Enter`** and follow the prompts to exit the file editor and save the script.

Using the GUI

1. Log in to the GUI.
2. Since the header editing scripts will be applied for all client requests through a cluster, the script must be configured on the cluster. Select the cluster or match rule on the left navigational pane and then select **Configuration > Header Editing** to display the **Header Editing** rule configuration screen.
3. Enter the scripts described above in the new braces beneath `client_request_edit()` and `server_response_edit()`.



```
client_request_edit ()
{
    $xf = header_value("X-Forwarded-For");
    $xf_save = header_value("X-Forwarded-For");
    str_insert($xf, "," . $CLIENT_IP, AFTER);
}

server_response_edit()
{
    header_insert("X-proxy-IP", $xf_save);
}
```

3. Click on the **Test** button to verify that the header edit script compiles successfully. If there is a problem with the script, you will be given the option to re-enter. Clicking on the **Reset** button will remove anything that you've entered and return the screen to the default display.
4. Click on **Commit** to save the script.

Appendix G

Maximum Configuration Values

The following table lists the allowable number of load balancing objects that can currently be defined on Equalizer:

	970LX	**670LX**	**470LX**	**370LX**	**EQOD**
TCP, UDP, HTTP Clusters	1000	1000	1000	500	425
HTTPS Clusters	200	200	150	150	125
Server Pools	250	190	125	95	80
Server Instances	521 Per Server Pool	375 Per Server Pool	250 Per Server Pool	190 Per Server Pool	150 Per Server Pool
Health Check Instances	16 Per Server Instance	16 Per Server Instance	16 Per Server Instance	16 Per Server Instance	16 Per Server Instance
VLANs	999	750	500	375	300
Subnets	999 Per VLAN	750 Per VLAN	500 Per VLAN	375 Per VLAN	300 Per VLAN
Envoy Resources	250	190	95	95	80
Total Objects	1200	900	450	450	380

Glossary

6

6in4

6in4 is an Internet transition mechanism for migrating from Internet Protocol version 4 (IPv4) to IPv6.

A

Access Control Lists (ACLs)

Refers to rules that are applied to port numbers or network daemon names that are available on a host or other layer 3, each with a list of hosts and/or networks permitted to use the service

active connection count

Shows the number of connections currently active on the server.

active connections weight

The relative influence on the policy of the number of active connections currently open to a server

Active Content Verification (ACV)

These are Server Health Checks. ACV health checks basically have Equalizer attempting to open a TCP connection to a server and determing if the server accepts. When the server accepts, Equalizer sends a user-defined request to the server (for example GET a web page). If the server does not accept the request within the configured time, or is the server does not respond with the response the user configured, the server is marked "down" and no further traffic is sent to that server until it starts responding to health checks.

active-active

Equalizer clusters are instantiated on two peers and organized into failover groups. If the other peer's connectivity for the failover group's resources is judged to be "healthier" than the peer on wihich the group is running, then the group fails over to the other peer.

active-standby

All Equalizer clusters are instantiated on a primary system, and all will failover to a backup unit if the backup unit is judgged to be "healthier" than the primary system.

ActiveX

ActiveX is a Microsoft software component for Windows. It controls are small programs, sometimes called add-ons that are used on the Internet and can enhance the browsing experience by allowing animation or helping with tasks such as Microsoft security updates.

adaptive load balancing

Distributes the load according to the following performance indicators for each server.

Address Resolution Protocol

Address Resolution Protocol (ARP) is a protocol used by IPv4, to map IP network addresses to the hardware addresses used by a data link protocol. The protocol operates below the network layer as a part of the interface between the OSI network and OSI link layer.

address translation

The modification of external addresses to standardized network addresses and of standardized network addresses to external addresses.

administration address
The IP address assigned to Equalizer on any VLAN. Access to Equalizer can be configured for each VLAN.

administration interface
The browser-based interface for setting up and managing Equalizer.

affinity
Affinity is a technique that enables the load balancer to remember which balanced server was chosen for a certain client at its initial request. Subsequent requests are then directed to the same server again. If the affinity feature is disabled when a new TCP/IP connection is received from a client, load balancer chooses the correct server at that moment and forwards the packet to it. If a subsequent connection comes in from the same client, load balancer treats it as an unrelated connection, and again chooses the most appropriate server at that moment.

Age (HTTP/1.1)
The age of a response is the time since it was sent by, or successfully validated with, the origin server.

agent
An application that gathers or processes information for a larger application. See server agent.

agent weight
The relative influence on the policy of the return value of a server agent (if any) running on the servers in the cluster.

aggregation
See link aggregation and sticky network aggregation.

algorithm
Instructions, procedures, or formulas used to solve a problem.

alias
A nickname that replaces a long name or one that is difficult to remember or spell.

aliased IP address
A nickname for an IP address.

Application Delivery Controller (ADC)
An application delivery controller (ADC) is a network device that usually sits between the firewall/router and application servers. The ADC is in many cases described as the next generation load balancer.

application layer
Layer 7 of the Open Systems Interconnection (OSI) network model, where communication between endpoints is defined by the application.

atom
The smallest part of a regular expression in Equalizer. See branch, piece, and regular expression.

authoritative name server
A name server that maintains the master records for a particular domain. See name server.

B

back-end server
A physical server that is part of a virtual cluster on Equalizer.

backup Equalizer
The backup unit in a failover pair of Equalizers. The backup unit constantly monitors the health of the active (primary) unit, and replaces the primary unit in the event that the primary becomes unavailable. See hot backup and primary Equalizer.

bound
A character that represents the limit of part of a regular expression.

bracket expression
In a regular expression, a list of characters enclosed in brackets ([...]).

branch
In an Equalizer regular expression, a complete piece of a regular expression. You can concatenate and/or match branches. See atom, piece, and regular expression.

C

cache
An area in which information is temporarily stored.

cacheable
A response is cacheable if a cache is allowed to store a copy of the response message for use in answering subsequent requests.

client timeout
The time in seconds that Equalizer waits before closing an idle client connection.

cluster
A set of networked computer systems that work together as one system. See server cluster and virtual cluster.

cluster address
The IP address assigned to a particular cluster configured on Equalizer.

command transfer
In a failover configuration the subnet that is the subnet over which the configuration file transfers (between preferred primary and preferred backup) can occur.

Community String
Any SNMP management console needs to send the correct community string along with all SNMP requests. If the sent community string is not correct, Equalizer discards the request and will not respond.

computed load
A measure of the performance of a server relative to the overall performance of the cluster of which the server is a part.

connection
A connection is a Layer 4 transmission path established between two endpoints. Clients open connections to Equalizer cluster IPs, and Equalizer opens connections to the servers behind it. The notion of a connection is supplied by the underlying protocol. There are connection-oriented protocols, like TCP and connectionless protocols, such as UDP.

connection timeout (ms)
This is the failover timeout (in ms) for peer connections.

cookie
Data that a Web server stores on a client on behalf of a Web site. When a user returns to the Web site, the server reads the cookie data on the client, providing the Web site all the saved information about the user.

cookie header

One of Equalizer's supported headers, a cookie header is an HTTP data string previously sent by a server that is stored in Equalizer for future routing.

cookie persistence

In cookie-based persistence, Equalizer "stuffs" a cookie into the server's response header on its way back to the client. This cookie uniquely identifies the server to which the client was just connected. The client includes (sends) the cookie in subsequent requests to the Equalizer. Equalizer uses the information in the cookie to route the requests back to the same server.

cookie switching

Refers to three distinct ways to perform cookie switching: cookie-read, cookie-insert, and cookie-rewrite.

D

daemon

An application that runs in the background and performs one or more actions when events trigger those actions.

default gateway

A default gateway is on the same subnet as Equalizer, and is the gateway which Equalier relies on to route traffic.

delay weight

The relative influence on the policy of the current response time between Equalizer and the server.

Direct Server Return (DSR)

In a Direct Server Return (DSR) configuration, the server receiving a client request responds directly to the client IP, bypassing Equalizer. Because Equalizer only processes incoming requests, cluster performance is dramatically improved when using DSR in high bandwidth applications, especially those that deliver a significant amount of streaming content. In such applications, it is not necessary for Equalizer to receive and examine the server’s responses: the client makes a request and the server simply streams a large amount of data to the client.

DNS

Domain Name System or Domain Name Service; used to map domain names to Internet servers in order to link to IP addresses or map IP addresses to domain names. See IP address.

DNS TTL

The amount of time, in seconds, that a name server is allowed to cache the domain information. See DNS and TTL.

domain

The highest level in an IP address and the last part of the address in the URL. The domain identifies the category under which the Web site operates. For example, in www.coyotepoint.com, com is the domain, where com represents a commercial site. See domain name, IP address, and subdomain. See also DNS.

domain name

The owner of an IP address. The next highest level in an IP address and the next-to-last part of the address. For example, in www.coyotepoint.com, coyotepoint is the domain name. See domain, IP address, and subdomain. See also DNS.

dual stack networking

Dual-stack networking is a transition technology in which IPv4 and IPv6 coexist and operate in tandem and independently over shared or dedicated links. In a dual-stack network, IPv4 and IPv6 are fully deployed across the network infrastructure so that configuration and routing protocols handle both IPv4 and IPv6 addressing.

dynamic weight

The weight that Equalizer assigns to a particular server during operation. See server weight, initial weight, and weight.

E

echo
An IP address-port pair that identifies the start or end of an address; a value that ends a process.

Enhanced NAT
NAT performed by a load balancer with protocol-specific knowledge in order to make certain protocols with with load balancing.

Envoy
Equalizer add-on software that supports geographic clustering and load balancing. See geographic cluster, geographic load balancing, and load balancing. See also intelligent load balancing.

eqcli
The Equalizer EQ/OS 10 Command Line Interface

EQOD
See "Equalizer OnDemand"

Equalizer Administration Interface
An Equalizer window with which you can monitor Equalizer's operation; view statistics; add, modify, or clusters; add, modify, and delete servers; and shut down a server or Equalizer through a Javascript-enabled browser.

Equalizer Configuration Utility
An Equalizer feature that enables you to configure Equalizer, set parameters, and shut down and upgrade Equalizer.

Equalizer OnDemand
Equalizer OnDemand is a software-based virtual appliance that operates as an integral part of the virtual infrastructure model. It is deployed as a single virtual server instance dedicated to load balancing and managing the application delivery needs.

external address
The IP address assigned to Equalizer on the external network.

external interface
A network interface used to connect Equalizer to the external network. See interface, internal interface, and network interface.

external network
The subnet to which the client machines and possibly the Internet or an intranet are connected.

F

failover
The act of transferring operations from a failing component to a backup component without interrupting processing.

firewall
A set of security programs, which is located at a network gateway server and which protect the network from any user on an external network. See gateway.

FQDN
See Fully Qualified Domain Name (FQDN).

fresh (HTTP/1.1)
A response is fresh if its age has not yet exceeded its freshness lifetime.

FTP

File Transfer Protocol; rules for transferring files from one computer to another.

FTP cluster

A virtual cluster providing service on the FTP control port (port 21). See cluster and virtual cluster.

Fully Qualified Domain Name (FQDN)

The complete, registered domain name of an Internet host, which is written relative to the root domain and unambiguously specifies a host’s location in the DNS hierarchy. For example, east is a hostname and east.coyotepoint.com is its fully qualified domain name. See also domain name.

G

gateway

A network route that typically translates information between two different protocols.

geographic cluster

A collection of servers (such as Web sites) that provide a common service over different physical locations. See cluster.

geographic load balancing

Distributing requests as equally as possible across servers in different physical locations. See load balancing. See also intelligent load balancing.

geographic probe

A query sent to a site in a geographic cluster to gather information so Equalizer can determine the site that is best able to process a pending request. See geographic cluster.

H

header

One or more lines of data that identify the beginning of a block of information or a file.

Health Check Weight

Health Check Weight is only associated with Load Checks The weight describes the extent to which the given load check's load value influences the calculation of the total load on the object to which it is attached.

High Availability

High Availability, or HA as it is commonly called, refers to the availability of resources in the wake of component failures in a load balancing configuration.

hot backup

Configuring a second Equalizer as a backup unit that will take over in case of failure. Also known as a hot spare. See backup Equalizer. See also primary Equalizer. A server can also be used as a hot backup, or hot spare, within a cluster. If all the other servers in the cluster fail, the hot spare will begin processing requests for the cluster.

HTTP

HyperText Transfer Protocol; the protocol with which a computer or user access information on the World Wide Web.

HTTPS

HyperText Transfer Protocol (Secure). The SSL/TLS protocol is used in combination with the HTTP protocol to provide secure identification and data encryption.

hub

A device that joins all the components attached to a network.

I

ICMP

Internet Control Message Protocol. Used by operating systems of networked computers to send error messages indicating that a requested service is not available or that a host or router could not be reached.

ICMP echo request

The act of repeating a stream of characters (for example, echoing on the computer screen characters as a user types those characters). See ping. See also echo.

ICMP Probe Maximum Tries

Enables probing servers using ICMP echo (ping) probes. These probes are 5 seconds apart. If a server does not respond to an ICMP prebend it has attempted to probe the number of times specified, it is marked down only if there are no other probes (TCP, ACV, or server agent) active for the cluster.

ICMP Probes

These are Server Health Checks. ICMP health checks basically have Equalizer sending a "ping" to a server and "listening" if the server sends a response. If the server does not respond within the configured time, the server is marked "down" and no further traffic is sent to that server until it starts responding to health checks.

ICMP triangulation

Routing client requests to the closest site geographically based on triangulation, a method of calculating the location of a site using the known locations of two or more other sites.

initial weight

The weight that an administrator assigns to a particular server. During operation, Equalizer dynamically adjusts the server weights (that is, dynamic weight), so a server's weight at a particular time might be different from the initial weight originally set by the administrator. See dynamic weight, server weight, and weight.

intelligent load balancing

A request for load balancing using Equalizer-based algorithms that assess the configuration options set for cluster and servers, real-time server status information, and information in the request itself. See algorithm and load balancing. See also geographic load balancing.

interface

The place at which two or more systems connect and communicate with each other. See external interface, internal interface, and network interface.

internal address

The IP address assigned to Equalizer on the internal network.

internal network

The subnet to which the back-end server machines are connected.

Internet Control Message Protocol (ICMP)

The ISO/OSI Layer 3, Network, protocol that controls transport routes, message handling, and message transfers during IP packet processing. See ICMP triangulation and ISO/OSI model.

IP

Internet protocol; the TCP/IP protocol that controls breaking up data messages into packets, sending the packets, and reforming the packets into their original data messages. See Internet protocol stack, IP address, packet, and TCP/IP.

IP address

A 32-bit address assigned to a host using TCP/IP. IP addresses are written in dotted decimal format, for example, 192.22.33.1.

IP Reputation Database

An IP Reputation database aggregates data from locations and sources around the world that collaborate to provide up to date information about threatening sources. It is commonly used as protection against malicious sources associated with web attacks, phishing activity, web scanning.

IPv4

Internet Protocol version 4 (IPv4) is the fourth revision in the development of the Internet Protocol (IP) and the first version of the protocol to be widely deployed. Together with IPv6, it is at the core of standards-based internetworking methods of the Internet.

IPv6

IPv6 (Internet Protocol version 6) is a version of the Internet Protocol (IP) intended to succeed IPv4, which is the protocol currently used to direct almost all Internet traffic.[1]

IPv6-IPv4 Translation

IPv6-IPv4 Translation is used when a portion of an internal network only supports IPv4 and/or a portion of applications can accept only IPv4 connections. The advantage of this is that IPv6 to IPv4 translations are provided for legacy IPv4 application services and provides additional time for completely migrating to IPv6 architecture.

ISO/IEC

International Organization for Standardization/International Electrotechnical Commission; international standards organizations.

ISO/OSI model

International Organization for Standardization/Open Systems Interconnection model, a standard that consists of seven layers that control how computers communicate with other computers over a network. Layer 1, Physical, which sets the rules for physical connections via hardware, is the lowest layer. Layer 2, Data-link, uses Layer 1 and its own rules to control coding, addressing, and transmitting information. Layer 3, Network, uses the prior two layers rules as well as its own rules to control transport routes, message handling, and message transfers. Layer 4, Transport, uses its rules and those of the previous layers to control accuracy of message delivery and service. Layer 5, Session, uses its rules and those of the previous layers to establish, maintain, and coordinate communication. Layer 6, Presentation, uses its rules and those of the previous layers to control text formatting and appearance as well as conversion of code. Layer 7, Application, uses its rules and those of the other layers to control transmission of information from one application to another. Layer 7 is the highest layer. See Layer 4, Layer 7, and transport layer.

L

L4

See Layer 4.

L4 Load Balancing

Layer 4 load balancing is the most basic form of load balancing. It is only aware of IP information present in UDP or TCP headers, that is IP addresses and TCP/UDP port numbers. You can load balance a great many applications using this capability, which functions as follows: A packet arrives at Equalizer with a specific destination IP address and port number. We look in the list of configured clusters to find one that matches. Attached to that cluster areservers.The "best server" is chosen and the packet/request is then sent to that server using NAT.

L4 TCP and UDP

IP protocols. They are well described at http://en.wikipedia.org/wiki/Transmission_Control_Protocol and http://en.wikipedia.org/wiki/User_Datagram_Protocol respectively. TCP is a way in which computers connected to a network can communicate reliably. TCP protects against data loss, corruption and reordering at the cost of some performance. TCP is the underlying protocol for HTTP, email and many common Internet applications. UDP is a more performant protocol which does not protect data from all the issues described above. It is however more useful for time-sensitive data so it is commonly used for audio, video and... DNS

L7

See Layer 7.

L7 Load Balancing

Layer 7 refers to the "application layer". That is, data embedded within the part of the data packet which is not TCP or IP header information. For example, in HTTP protocol, you can find information such as a requested URL or the data type in the L7 information. Equalizer can search for this information in the packets exchanged between clients and servers and make decisions about how to handle the traffic such as with Match Rules, Cookie Persistence, or Responders. Note that to this date Equalizer supports L7 load balancing for HTTP and HTTPS protocols.

LAN

See Local Area Network.

latency

The time over which a signal travels over a network, from the starting point to the endpoint. See ping. See also CMP echo request and echo.

Layer 4 (L4)

The transport layer; Layer 4 uses its rules and those of the previous three layers to control accuracy of message delivery and service.which controls accuracy of message delivery and service. See ISO/OSI model and Layer 7.

Layer 7 (L7)

The application layer; Layer 7 uses its rules and those of the other layers to control transmission of information from one application to another. Layer 7 is the highest layer in the ISO/OSI model. See ISO/OSI model and Layer 4.

least cxns (least connections)load balancing

Dispatches the highest percentage of requests to the server with the least number of active connections. In the same way as Fastest Response, Equalizer tries to avoid overloading the server so it checks the server’s response time and server agent value. Least Connections optimizes the balance of connections to servers in the cluster.

Link Load Balancing

Load balancing is a computer networking method for distributing workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units or disk drives.

Liveness Check

A Liveness Check is a health check that reports or is utilized only to assess UP/DOWN status. L3 (Ping), L4 (UDP and TCP), and ACV are liveness checks.

load

A job that can be processed or transported once. See load balancing. See also geographic load balancing and intelligent load balancing.

load balancing

Moving a load from a highly-used resource to a resource that is used less often so that operations are efficient. Equalizer balances loads over a wide physical area or by using algorithms that assess options and real-time information. See geographic load balancing and intelligent load balancing.

Load Check

A Load Check is a health check that reports a load value, and may also (directly or indirectly) indicate UP/DOWN status. VLB and Server Agent are types of Load Checks.

Local Area Network (LAN)

Local Area Network

M

Match Rules

Match Rules are a feature associated with Layer 7 load balancing. This is often called "content switching" by other vendors. Using match rules you can configure Equalizer to handle application (HTTP) requests in specific ways depending on the data present in the request. For example, you can tell Equalizer to send any image requests to one of three dedicated image servers while all other requests go to an application server. You can specify that if a page is requested which is company-internal only and the client is not on the local network to drop the request (or hand out a denied response).

Multi-gateway

A dedicated 0/0 gateway for every IP network defined on Equalizer

Multi-netting

Adding multiple layer 3 IP networks, to a single layer 2 environment (vlan, interface)

MX exchanger

Mail exchanger; a fully qualified domain name to be returned if a server receives a mail exchanger request.

N

N+1

Equalizer clusters are instantiated on all "N" peers and organized into failover groups. If the passive, or backup peer's connectivity for a failover group's resources is judged to be "healthier" that the peer on which the group is running, then the group fails over to the passive peer, which becomes the Primary peer.

name server

A server that stores information about the domain name space.

NAT

Network Address Translation; an Internet standard that defines the process of converting IP addresses on a local-area network to Internet IP addresses. See NAT subsystem.

NAT subsystem

The Equalizer subsystem responsible for transferring connections to and from the back-end servers.

netmask

Address mask; a bit mask used to select bits from an Internet address for subnet addressing. The mask is 32 bits long and selects the network portion of the Internet address and one or more bits of the local portion.

Network Address Translation (NAT)

See NAT.

network interface

The place at which two or more networks connect and communicate with each other. See interface. See external interface, interface, and internal interface.

network route

See gateway.

O

OSI network

A network that uses the International Organization for Standardization/Open Systems Interconnection model. See ISO/OSI model, Layer 4, Layer 7, and transport layer.

outbound NAT

If a client or server is using Equalizer as a gateway device, their source IP will translated to an Equalizer IP on the egress interface of Equalizer used to reach the destination network

P

packet

A group of data that is transmitted as a single entity.

passive FTP connection

An Equalizer option that rewrites outgoing FTP PASV control messages from the servers so that they contain the IP address of the virtual cluster rather than that of the server. See FTP and PASV.

PASV

Passive mode FTP; a mode with which you can establish FTP connections for clients that are behind firewalls. See firewall, FTP, and passive FTP connections.

pattern match

A pattern of ASCII or hexadecimal data that filters data.

payload

The set of data to be transmitted. A payload contains user information, user overhead information, and other information that a user requests. A payload does not include system overhead information. Also known as the mission bit stream.

persistence

The act of storing or retaining data for use at a later time, especially data that shows the state of the network before processing resumes. See cookie and IP-address-based persistence.

Persistence

Often, when a client (web browser) connects to an application, there is some "shared state" between the client and server which cannot be used with any other server. Using the following example: You point your web browser at www.website.com and log in. The server notes that you are logged in and allows you to use the application. Now,for example, there were two servers and a load balancer, and that the load balancer just alternated handing any requests it recieved between the two servers... You've been connected to server "A" where you log in. You then send a different request (run a report) and the load balancer sends your request to server "B". Since you are not logged into server "B" so you get an error and are asked to log in again. This is because you have NO PERSISTENCE. Your connection does not get connected to the same server each time. What you need is some way for the load balancer to recognize your browser and always send your requests to server "A". We do this in two ways. #1 is called "IP persistence" or "sticky persistence". The load balancer remembers your IP address the first time you connect to a server and records which server you were connected to. The next time you connect, Equalizer looks up your IP address in an internal table and notes that you belong on server "A", so that is where you are sent. Sticky Persistence is available with both L4 and L7 load balancing. An alternate persistence scheme is available under L7 only. It is called "cookie persistence" and it uses information stored at the client (a cookie http://en.wikipedia.org/wiki/HTTP_cookie) to recall which server it should be connected to. The advantage of keeping the persistence information on the client is that no memory is used on the Eq. This way millions of clients can have poersistence for an indefinate period of time. With sticky persistence the number of persistent clients is limited by the memory available to store the sticky table.

physical server

A machine located on the internal network that provides services on specific IP addresses and ports. See server and virtual web server. See also authoritative name server, back-end server, name server, and proxy server.

piece
An atom followed by a single *, +, or ?, or by a bound. See atom, branch, and regular expression.

ping
A program used to test reachability of destinations by sending them an ICMP echo request and waiting for a reply. See echo and probe. See also CMP echo request

port
The abstraction used by Internet transport protocols to distinguish among multiple simultaneous connections to a single destination host.

port grouping
Refers to the configuration of a load balancers with a list of application ports that must be treated as one group.

port number
The number used to identify a service contact port, such as HTTP port 80.

Port-Address Translation (PAT)
PAT is inherent in load balancers and refers to tranlsating the port number in TCP/UDP packets.

primary Equalizer
The primary unit that handles requests. If the primary Equalizer fails, the backup unit replaces it. See also backup Equalizer and hot backup.

probe
An action that obtains status information about a computer, network, or device. See geographic probe and ping.

probing interval
The target interval between TCP probes of a cluster that has been marked failing in the load balancing daemon's internal tables. If the server does not respond to strikeout threshold (see below) additional TCP probes after it is marked failing, then the server is marked down. These additional probes are at least probe interval seconds apart. This value is solely a target; the monitoring process adjusts itself based on a number of factors, including system load. The default value is 20 seconds.

protocol
A set of rules that govern adherence to a set of standards. See protocol stack.

protocol stack
A layer of protocols that process network actions cooperatively and in tandem. See protocol.

proxy server
A utility, which is part of a firewall, that helps the regular tasks of managing data transmittal from a network to the Internet and from the Internet to the network. See also firewall.

Q

quiesce
Quiesce is a server option that, when enabled, tells Equalizer not to send the server any new traffic, while existing connections are allowed to complete. Eventually, the server should not be serving any traffic. Sometimes called 'server draining'.

R

RADIUS
Remote Authentication Dial-In User Service; a protocol that authorizes and authenticates a user trying to link to a network or the Internet.

receive timeout

This is the failover timeout (in ms) for receiving peer data.

redirection

The process of receiving input from or sending output to a different resource than usual.

regular expression (RE)

One or more non-empty branches, separated by pipe symbols (|). An expression matches anything that matches one of the branches. See atom, branch and piece.

request packet

A packet that contains information that requests a response. See packet and response packet.

reserved network

A network consisting of “phony” IP addresses, which are not registered and cannot be made visible outside of the internal network.

resolution

The process of interpreting all the messages between an IP address and a domain name address.

Resource

A Resource is any object that is being probed by a health check.

Responder

This is an L7 advanced feature. A user can configure Equalizer to send a response directly to the client under specified circumstances. Without involving a server. Responders come in two flavors. "Sorry"

response load balancing

Dispatches the highest percentage of requests to the server with the shortest response time. Equalizer does this carefully: if Equalizer sends too many requests to a server, the result can be an overloaded server with slower response time. The fastest response policy optimizes the cluster-wide response time. The fastest response policy also checks the number of active connections and server agent values (if configured); but both of these have less of an influence than they do under the adaptive load balancing policy. For example, if a server’s active connection count and server agent values are high, Equalizer might not dispatch new requests to that server even if that server’s response time is the fastest in the cluster.

response packet

A packet that contains information that responds to a request. See packet and request packet.

retry interval (ms)

This is the time (in ms) between failed failover peer probes.

round robin

The default load balancing policy which distributes requests equally among all servers in a virtual cluster, without regard to initial weights or adaptive load balancing criteria. The first request received is routed to the first server in the list, the second request to the second server, and so on. When the last server is reached, the cycle starts again with the first server.

round-robin load balancing

Diistributes requests equally among all the servers in the cluster. Equalizer dispatches the first incoming request to the first server, the second to the second server, and so on. When Equalizer reaches the last server, it repeats the cycle. If a server in the cluster is down, Equalizer does not send requests to that server. This is the default method.

router

A network device that facilitates the transmission (that is, routing) of messages.

routing table

A database, which is static or dynamic, that contains a set of route addresses and routing information familiar to the router. A human being enters and updates the information in a static routing table; routers operate and constantly update a dynamic routing table.

RST

Refers to the TCP protocol's reset command, which instructs a device to end a connection.

S

Secure Sockets Layer (SSL)

A protocol that enables secure communication between two hosts, using data encryption and authentication.

server

A computer or application that controls access to a network and its associated devices and applications. A server communicates with one or more clients as well as other servers. See authoritative name server, back-end server, name server, physical server, proxy server, and virtual web server.

server address

The IP address of a server on the internal interface. Multiple IP addresses can be aliased to a single physical server. See server.

server agent

An agent that provides Equalizer with real-time performance statistics for a specified server. See server.

server agent load balancing

Dispatches the highest percentage of requests to the server with the lowest server agent value. In a similar way to Fastest Response, Equalizer tries to avoid overloading the server by checking the number of connections and response time. This method only works if server agents are running on all servers in the cluster.

server agent value

The value returned by the server agent daemon (if any) running on the server.

Server Agents

These are somtimes called Smple Health Checks. Simple health checks resemble ICMP, TCP and ACV server health checks, but they have a slightly different purpose. Rather than determining if a server is either "up" or "down", they send a request to a server and the server is expected to reply with a number between -1 and 100 to say exactly how alive it is. In order to give Equalizer a proper answer to the health check query, the server needs to run a Server Agent.

server cluster

A group of servers that are components in a network and joined through hardware or software. See cluster. See also FTP cluster, geographic cluster, and virtual cluster. See server.

server draining

The process of allowing existing connections to a server to complete while not allowing any new connections, so that the server is eventually not serveing any traffic. Usually done in preparation for shutting down, rebooting, or upgrading a server. On Equalizer, server draining is enabled using the quiesce server option.

server endpoint

An IP address-port pair that identifies a physical or virtual server on the internal network to which Equalizer can route connection requests. See server.

Server Name Indication (SNI)

Server Name Indication (SNI) is an extension to the SSL and TLS protocols that indicates a server name or website that a client is attempting to connect with at the start of the handshake process. It allows a server to present multiple certificates on the same IP address and port number, thus allowing multiple secure (HTTPS) websites to be server of the same IP address while allowing all of those sites to have unique certificates all serviced on the same cluster/IP address.

server response time
The length of time for the server to begin sending reply packets after Equalizer sends a request.

Server Side Encryption
Server Side Encryption (SSE) provides you with the ability to configure a cluster and/or match rule so that traffic between Equalizer and back end servers is encrypted using SSL/TLS, eliminating the untrusted paths.

server weight
A value that indicates the relative proportion of connection requests that a particular server will receive. See dynamic weight, server, initial weight, and weight.

session
A logical connection between a server and a client that may span a series of individual client requests and server responses (i.e., transactions). Depending on the application, a session may also span multiple client-server connections as well as transactions. Session data is typically maintained using cookies inserted into client requests and server responses, by Equalizer, servers, or both. Session data may also be maintained on clients and servers. Equalizer uses cookies at Layer 7 and a sticky timer at Layer 4 to provide server persistence; the cookie lifetime or sticky time to set on Equalizer is determined by the application, and should usually match the corresponding cookie or session timeouts set on the real servers in a cluster.

SFP and SFP+
SFP+ is the new version which supports 10Gb throughput. Stands for "Small Factor Pluggable transceiver". A port with which a number of different network cables to be used, including regular 'copper' cables, or 'fiber' (optical) cables.Fiber-optic uses a glass cable to send light signals, and copper users a metal (copper) cable to send electrical signals.

site
An Envoy site is part of an Envoy geocluster. It points to an existing virtual cluster on an Equalizer running Envoy.

source IP-based persistence
When a TCP SYN packet is received, a load balancer looks for hte source IP address in its session table. If an entry is not found, it treats the user as "new" and selects a server based on the load balancing algorithm. to forward the packet. The load balancer also makes an entry in the session table. If an entry for this source IP address is found in the session table, the load balancer forwards the packet to the same server that received the previous connection for this source IP address regardless of the load balancing algorithm.

Source NAT
This inbound address translation feature allows Equalizer to substitute one of it's own IP addresses for the requesting clients IP address.

Source Routing
Source routing allows a sender of a packet to partially or completely specify the route the packet takes through the network. In contrast, in non-source routing protocols, routers in the network determine the path based on the packet's destination.

spoofing
Using the client’s IP address for the source IP address in client requests. This fools (or spoofs) the server into regarding the client as the source of the request. For spoofing to work, the default gateway for the server must be set to Equalizer’s internal IP address.

SSL
See Secure Sockets Layer (SSL).

stack
An area of reserved memory in which applications place status data and other data. See protocol stack.

stale (HTTP/1.1)
A response is stale if its age has passed its freshness lifetime.

stale connection

A partially open or closed connection.

state

Status; the current condition of a network, computer, or peripherals.

stateless

A condition in which a server processes each request from a site independently and cannot store information about prior requests from that site. Each request stands on its own. See also DNS and RADIUS.

sticky connection

A Layer 4 connection in which a particular client remains connected to same server to handle subsequent requests within a set period of time. Sticky connections are managed on Equalizer using sticky records, which record the server-client connection details; sticky records expire according to the configured sticky timer setting.

sticky network aggregation

Basically, this is server affinity determined by a network mask at Layer 4. If the following conditions are all true: an incoming request to a Layer 4 cluster has a source IP that matches the sticky network mask set for the cluster the destination port on the server is not responding the same server IP with another port is defined in another cluster Then Equalizer will attempt to forward the request to the same server on the other port.

sticky timer

A countdown timer used to manage sticky connections to a Layer 4 cluster. When this timer expires (i.e., there is no activity between the server and client for the duration of the timer setting), Equalizer removes the sticky record for the connection.

strike count

This is the maximum number of failed failover peer probes. The global strike count is not used until ALL heartbeating subnets have at least one strike. Once each subnet has a strike, we fail over when the number of failed heartbeats across all heartbeating subnets is equal to or greater than the global strike count

subdomain

A section, which is formally named, that is under a domain name; analogous to the relationship between a subfolder and folder. For example, in www.coyotepoint.com, www is the subdomain. See domain, domain name, and IP address. See also DNS.

subnet

Part of a network that has the same address as the network plus a unique subnet mask.

switch

A Layer 2 device that connects network segments into one broadcast domain.

SYN/ACK

Synchronize and acknowledge; a message that synchronizes a sequence of data information and acknowledges the reception of that information.

syslog

A system log file, in which information, warning, and error messages are stored in a file, sent to a system, or printed.

System Contact

Contact is the name of the person responsible for this unit.

System Descriptions

The user-assigned description of the Equalizer.

System Location

Location describes Equalizer's physical location.

System Name

The name assigned to the system. By default it is Equalizer.

T

TCP

Transmission Control Protocol; the rules for the conversion of data messages into packets. TCP providesSee ISO/OSI model, Layer 4, packet, transport layer.

TCP Probes

These are Server Health Checks. TCP health checks basically have Equalizer attempting to open a TCP connection to a server and determining if the server accepts. If the server does not accept within the configured time, the server is marked "down" and no further traffic is sent to that server until it starts responding to health checks.

TCP/IP

Transmission Control Protocol/Internet Protocol; the rules for transmitting data over networks and the Internet.

Telnet

Part of TCP/IP, a protocol that enables a user to log onto a remote computer connected to the Internet. See TCP/IP.

traceroute

A utility that shows the route over which a packet travels to reach its destination.

transaction

A transaction is a Layer 7 interaction between a client and a server over a network protocol that defines the format of the interaction. An HTTP transaction, for example, is a single client request and associated server response. HTTP is thus a transaction-oriented protocol. This is in contrast to Layer 4 TCP, which is a connection-oriented protocol and does not provide the notion of a transaction to applications running over it.

Transmission Control Protocol (TCP)

See TCP.

Transmission Control Protocol/Internet Protocol (TCP/IP)

See TCP/IP.

transport layer

See Layer 4. See also ISO/OSI model.

TTL

Time-to-live, the length of time, in seconds, that a client’s DNS server should cache a resolved IP address.

U

User Datagram Protocol (UDP)

Within TCP/IP, a protocol that is similar to Layer 4 (the transport layer). UDP converts data into packets to be sent from one server to another but does not verify the validity of the data. See ISO/OSI, TCP/IP, and transport layer.

V

virtual cluster

An endpoint that acts as the network-visible port for a set of hidden back-end servers. See cluster, endpoint, FTP cluster, geographic cluster, and server cluster.

virtual server address
An IP address that is aliased to a physical server that has its own, separate IP address. See virtual web server.

virtual web server
Software that imitates HTTP server hardware. A virtual web server has its own domain name and IP address. See domain name, HTTP, IP address, server, and virtual server address. See also authoritative name server, back-end server, name server, physical server, and proxy server.

VLAN
See Virtual Local Area Network.

VM CPU
For servers that are associated with VMware Virtual Machines, the relative influence on the policy of the VM CPU usage status returned by VMware.

VM RAM
For servers that are associated with VMware Virtual Machines, the relative influence on the policy of the VM RAM usage status returned by VMware.

W

WAP
See Wireless Application Protocol.

Web Application Firewall
A web application firewall (WAF) is an appliance, server plugin, or filter that applies a set of rules to an HTTP conversation. Generally, these rules cover common attacks such as cross-site scripting (XSS) and SQL injection. By customizing the rules to your application, many attacks can be identified and blocked.

weight
The relative proportion of a single item in a population of similar items. See dynamic weight, server weight, and initial weight.

Wireless Application Protocol (WAP)
A set of rules that govern access to the Internet through wireless devices such as cellular telephones, pagers, and two-way communication devices.