Cisco BroadWorks

Xtended Services Platform Load Balancing

Feature Description

Release 22.0

Document Version 2.16

XspLoadBalancingFD-R220

Market Request Number 4110

Feature Request Number 2305

Copyright Notice

Trademarks

Document Revision History

Version	Reason for Change	Date
0.1 Draft	Created document.	October 27, 2015
0.2 Draft	Completed sections *9 Provisioning Interface Impacts* through to the end.	December 9, 2015
0.3 Draft	Prepared document for FS review.	December 11, 2015
0.4 Draft	Incorporated comments from FS review.	December 18, 2015
2.0 Approved	Approved document.	December 21, 2015
2.1	Incorporated additional *WebContainer* and *LogServer* impacts.	January 19, 2016
2.2	Corrected typographical errors in performance counter descriptions.	January 26, 2016
2.3	Modified description of configuration reload functionality and associated provisioning. Added server weight provisioning. Added more logging information.	February 5, 2016
2.4 Approved	Incorporated comments from FS re-review.	February 5, 2016
2.5	Received approval for *Install Changes*.	February 8, 2016
2.6	Added default backend description. Updated provisioning default values. Corrected CLI impacts.	February 8, 2016
2.7	Added SNI feature operation alias and CLI impacts.	February 25, 2016
2.8 Approved	Reapproved document.	March 1, 2016
2.9	Edited document.	March 4, 2016
2.10	Approved editing changes.	March 10, 2016
2.11	Finalized document but did not publish it.	March 10, 2016
2.12	Updated third-party software version.	April 1, 2016
2.13	Added new counters.	April 7, 2016
2.14	Edited changes and prepared document for publishing.	September 22, 2016
2.15	Added keywords to document properties in preparation for migration to Cisco.	August 17, 2020
2.16	Completed latest rebranding for Cisco and republished document.	September 4, 2020

Table of Contents

1 Feature Purpose and Overview

Applicable Telephony Application Server (TAS)
Not applicable

1.1 Purpose

Xtended Services Platform (Xsp) load balancing is a feature allowing easier management of dynamic Xtended Services Platform instantiation when Xtended Services Platforms are frequently added and removed from a farm. Previous techniques used the DNS entries to allow clients to discover new nodes. This process is too slow for rapidly evolving networks since it can take a long time for DNS caches to be refreshed.

With this new load balancing capability, clients always connect to a load balancing node before being routed to the actual backend Xtended Services Platform. The load balancing node can be reconfigured on the fly to add or remove Xtended Services Platforms.

1.2 Overview

The Load Balancer is a new Cisco BroadWorks application that can be deployed on an Xtended Services Platform. It listens for connections on TCP sockets and receives HTTP requests from clients. It then parses the request and determines the correct destination server to which the request is routed. When the response is received, it is transferred back to the client.

The Load Balancer's configuration includes servers to which requests are forwarded. These servers can be grouped together according to certain common capabilities. Therefore, servers that host the same web applications or perform the same functionality can be grouped together. This allows the Load Balancer to automatically distribute the load among the servers in a server group.

The Load Balancer also monitors the health of individual servers. When they fail to respond to an HTTP request, they are removed from the load balancing pool and the remaining servers from that pool continue to serve requests as usual. Periodically, the Load Balancer tests the failed servers and reintroduces them to the load balancing pool once they return to a healthy state.

Since the Load Balancer must read the HTTP request to route it correctly, the Load Balancer is also a Secure Sockets Layer (SSL) terminator. It therefore implements the same security and certificate management facility as the one currently employed on the Web Container application.

2 Use Cases

2.1 Provisioning Use Cases

2.1.1 Administrator Configures Load Balancer Proxy Servers, Backends, and Backend Servers

The administrator configures the Load Balancer by first enabling the desired network interfaces and ports. The administrator then defines backend server groups. Each group is defined so that each member of that group can perform the same tasks as the other members. The administrator adds servers to each group by specifying a name, address, port, and whether the connection is secure.

Once the backends and proxy servers are defined, the administrator determines which backend serves which web application, and the administrator configures the rules that bind proxy servers to backends accordingly.

2.1.2 Administrator Configures Load Balancer Proxy URLs and Bindings

The administrator wants to configure the Load Balancer to serve two applications that are hosted on two separate Xtended Services Interface farms, such as Device Management and Xtended Services Interface (Xsi). The administrator creates two proxy servers listening on the Load Balancer's public IP. The first one listens on Port 80 and is unsecured while the second one listens on Port 443 and is secure.

The administrator then creates four backend server groups.

Backend	URLs	Servers
XSI-unsecured	*/xsi-actions* */xsi-events*	XSP1:80 unsecured XSP2:80 unsecured
DeviceManagement-unsecured	*/dms*	XSP3:80 unsecured XSP4:80 unsecured
XSI-secured	*/xsi-actions* */xsi-events*	XSP1:443 secured XSP2:443 secured
DeviceManagement-secured	*/dms*	XSP3:443 secured XSP4:443 secured

Finally, the administrator configures the bindings between the proxy servers and the backend server groups. Four bindings are created.

Proxy Server	Backend
:80	*XSI-unsecured*
:80	*DeviceManagement-unsecured*
:443	*XSI-secured*
:443	*DeviceManagement-secured*

2.1.3 Administrator Modifies Backend Configuration

The administrator adds or removes a server from the configuration of a backend. After the configured time period, the Load Balancer reloads its configuration. The Load Balancer continues to serve previous connections using the old configuration while the new configuration is loaded. New clients connecting to the Load Balancer are routed to the new backend servers.

2.2 Feature Operation Use Cases

2.2.1 User Accesses Web Portal Through Load Balancer

The user accesses the web portal normally, using the same URL as before. The Load Balancer inspects the HTTP request. The headers are read and the correct backend is chosen. The Load Balancer uses the configured balancing algorithm to determine the destination server among all valid backend servers and forwards it the request. The backend server receives the HTTP request, processes it, and sends back the HTTP response. The Load Balancer inspects the HTTP response. A cookie indicating the backend server that processed the request is added to the list of headers. The response is forwarded back to the user.

2.2.2 User Accesses Web Portal Once Again

The user sends another HTTP request to the web portal to the same web application. When the Load Balancer receives the request, it is inspected. The backend is selected according to the binding rules. A routing cookie is found in the HTTP request that identifies the server to select. The Load Balancer selects the server identified by the cookie and forwards it the request.

2.2.3 Backend Server Is Removed from Load Balancing Pool

The user sends an HTTP request through the Load Balancer. The Load Balancer forwards it to a backend server, but after a configurable amount of time, no response is received. The Load Balancer sends the request to a different server from the same backend and temporarily removes the unresponsive server from the list of available servers. The request is successfully processed by the new server and the response is returned to the user.

2.2.4 Backend Server Is Added to Load Balancing Pool

While a backend server has been found unresponsive, the Load Balancer periodically attempts to open an HTTP connection to it to attest its health status. The check consists of an HTTP request to a configurable URL.

When the Load Balancer successfully receives a response containing one of the valid configurable response codes, the server is put back into the list of available servers, and new user requests can be forwarded to it.

2.2.5 Server Affinity

2.2.5.1 URL Server Affinity

An application generates a single URL and shares it with two separate clients. This URL identifies a resource that must be accessed on the same backend server. Therefore, the backend is configured with the URL server affinity scheme.

The clients then connect to that same URL through the Load Balancer. The Load Balancer inspects the request from the first client. The URL is used to identify the backend server and the request is forwarded to that server. When the Load Balancer inspects the request from the second client, the same processing is completed. Since this request contains the exact same URL as the first request, it is forwarded to the same backend server.

2.2.5.2 URL Parameter Server Affinity

An application generates URLs and shares them with two separate clients. These URLs contain a parameter that identifies a resource that must be accessed on the same backend server. Therefore, the backend is configured with the URL parameter server affinity scheme.

The clients then connect to their respective URL through the Load Balancer. The Load Balancer inspects the request from the first client. The URL's parameters are extracted. The parameter whose name matches the configured name is used to identify the backend server, and the request is forwarded to that server. When the Load Balancer inspects the request from the second client, the same processing is completed. Since that request contains the exact same URL parameter as the first request, it is forwarded to the same backend server.

2.2.6 Backend Reconfiguration

During operation of a system, the administrator is alerted that one of the Xtended Services Interface farms is experiencing an overload condition. To alleviate the condition, a new Xtended Services Platform is brought online. This new Xtended Services Platform is added to the existing farm. In the Load Balancer, that Xtended Services Platform is added to the list of servers inside the affected backend. To minimize downtime, the administrator initiates a soft reload of the configuration of the Load Balancer from the command line interface (CLI).

3 Provisioning Description

The following graph shows the different interactions between the configuration concepts. The Load Balancer is composed of multiple proxy servers. Each proxy server is associated with one or many backends through mappings called bindings. Backends are configured to contain references to backend servers hosting a relevant application. It is possible for several backends to reference the same server.

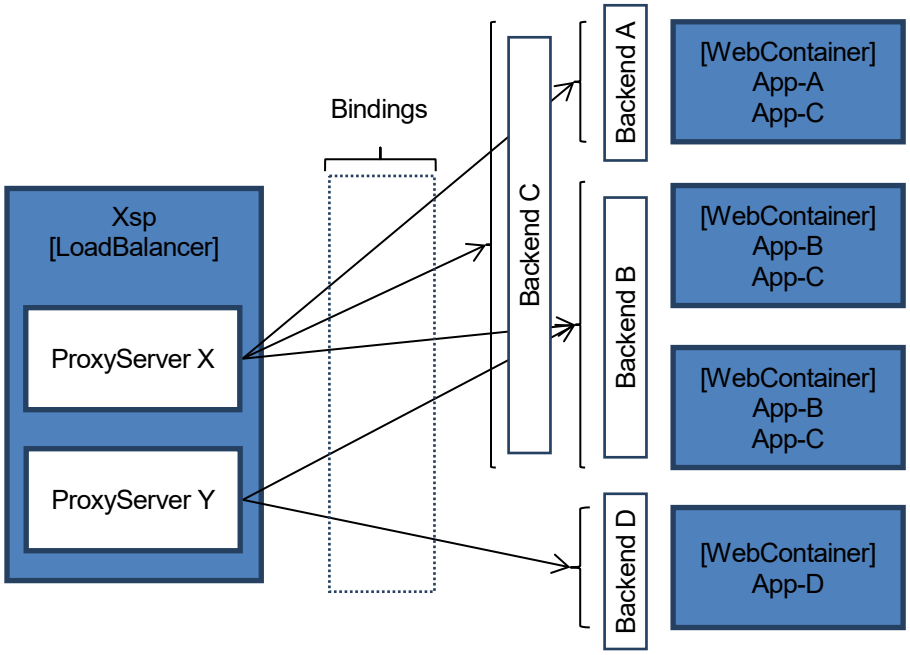


Figure 1 Representation of Interactions Between Configuration Concepts

3.1 General Settings

Name	Description
NbProc	This is the number of Load Balancer processes that are started.
MaxConn	This is the maximum number of incoming and outgoing connections that the Load Balancer accepts.
MaxQueuedConnections	This is the size of the listen backlog. When the maximum number of connections is reached, connections stay in the backlog until they can be accepted.
ConnectTimeout	This is the maximum amount of time that the Load Balancer waits for a connection to a backend server to be established.
KeepAliveTimeout	This is the inactivity time-out after which the Load Balancer closes a connection.
QueueTimeout	This is the amount of time a request waits in the Load Balancer if it cannot be proxied right away.
HealthCheckInterval	This is the amount of time between two health checks.

3.2 Routing

The Load Balancer must define both frontend and backend interfaces to support the rich routing behavior that is desired. The frontend is defined similarly as the way HTTP interfaces are currently defined.

3.2.1 Proxy Server

Name	Description
Address	This is the IP address of the interface to which clients connect.
Port	This is the TCP port of the interface to which clients connect.
Name	This is a unique name identifying this proxy server.
Secure	This indicates whether this proxy server is secure.
Client Auth Req	This indicates whether this proxy server must authenticate the clients.
Ciphers	This is the list of supported ciphers for this proxy server.
Protocols	This is the list of supported Transport Layer Security (TLS) protocols for this proxy server.

The backend servers to which the traffic is forwarded are defined in their own collection. The servers are grouped together.

3.2.2 Backends

Name	Description
Name	This is a name identifying this backend.
Servers	This is a list of individual servers that make up this group.
Servers → Host	This is the host name or IP address of the server.
Servers → Port	This is the TCP port of the server.
Servers → Secure	This indicates whether this server is secure.
Servers → Weight	This is the relative weight of this server, used for load balancing among the other members of the group.
URLs	This is a list of URL prefixes that map to this group of servers.
BalancingPolicy	This specifies which load balancing policy to use from the following: round-robin, least connected.
SessionAffinity	This specifies how the Load Balancer preserves sessions across multiple requests. This can be either “cookie”, “url”, or “url parameter”. It is also possible to disable session affinity altogether. When “url” or “url parameter” is selected, the round-robin balancing policy is used even if it was configured otherwise.
CookieName	When using “cookie” session affinity, this specifies the name of the cookie to use.
UrlParameter	When using “url parameter” session affinity, this specifies the name of the URL parameter to use.

3.2.3 Bindings

The bindings allow for specifying which backend server groups are available for each individual proxy server. When no bindings are defined, all backend server groups are available from all proxy servers.

Name	Description
ProxyServerName	This is the name of the proxy server used for this binding.
BackendName	This is the name of the backend used for this binding.

3.3 Logging

The logs generated from the Load Balancer application are sent to an auxiliary application that processes and writes them to the platform's logging facility. It is possible to configure the logging for both of these applications.

3.3.1 Load Balancer Logging

3.3.1.1 General Settings

Name	Description
Enabled	This turns the logging on and off.
Severity	This specifies the default logging severity level.
Priority	This specifies the priority of the logging thread.
MaxQueueSize	This specifies the maximum size of the logging queue.
ShowThreadName	This enables thread name logging.

3.3.1.2 Input Channels

Name	Description
Generic	This input channel logs the generic logs of the application.

3.3.1.3 Output Channels

Name	Description
Stdout	This output channel logs to the standard output.
File	This output channel logs to a file.

3.3.1.4 Access Logs

Name	Description
AccessLog	This logs the HTTP requests and responses to its own separate file.

3.3.2 Load Balancer Controller Logging

3.3.2.1 General Settings

Name	Description
Enabled	This turns the logging on and off.
Severity	This specifies the default logging severity level.
Priority	This specifies the priority of the logging thread.
MaxQueueSize	This specifies the maximum size of the logging queue.
ShowThreadName	This enables thread name logging.

3.3.2.2 Input Channels

Name	Description
Generic	This input channel logs the generic logs of the application.
BCCT	This input channel is used by the BroadWorks Common Communication Transport (BCCT) service.
BCCT keep-alive	This input channel is used by the BCCT keep-alive service.
ServiceOS	This input channel is used by the ServiceOS.
SMAP	This input channel is used by the Software Management Application Protocol (SMAP) service.
NameService	This input channel is used by the name service.
GcLog	This input channel is used by the Java Virtual Machine (JVM) garbage collector.

3.3.2.3 Output Channels

Name	Description
Stdout	This output channel logs to the standard output.
File	This output channel logs to a file.

4 Feature Operation

4.1 Proxy Servers

The entry points of the Load Balancer are defined as a list of proxy servers. These proxy servers are represented as an IP address and a port. When the Load Balancer is started, it listens on these interfaces for incoming connections from clients.

4.1.1 Client Authentication

On secure interfaces, it is possible to enable client authentication. Client authentication allows the Load Balancer to verify the authenticity of clients connecting to the secure interface. This is done by adding trusts to the interface and associating certificate authorities with them.

4.2 Backends

When a request is received on a proxy server, the routing rules dictate on which backend the request is being forwarded. Backends are a group of request processing servers that contain the same resources, such as web applications.

4.2.1 Server Selection

When the Load Balancer routes a request to a backend, it selects from among the group of servers the server that will eventually serve the request according to the selected load balancing algorithm.

4.2.2 Backend Selection

Backends serve specific URLs as defined in their configuration. These URLs are used by the Load Balancer to select which backend processes a request. The URLs represent the prefix of the request URL's path. A backend can serve multiple URLs.

4.2.3 Backend URL Paths

Backend URL paths prefixes are used to match against HTTP request targets. The paths always start with a slash ("/") and can contain an arbitrary number of segments, separated by slashes. Paths are compared in terms of segment with the HTTP request target in search of a match.

For example, an HTTP request to the URL, *http://broadsoft.com/webapp/testing*, matches the backend URL path, */webapp*, but not the backend URL path, */webapp/test*.

If an HTTP request matches multiple backends, the most specific backend is chosen.

For example, an HTTP request to the URL, *http://broadsoft.com/webapp/testing*, matches the backend URL path, */webapp*, and also the backend URL path, "/". Since the path, */webapp*, is more specific than "/", the backend associated with "/webapp" is chosen.

If there are multiple backends using the exact same URL path and accessible from the same interface, the backend chosen by the Load Balancer is undefined.

4.2.4 Default Backend

A default backend is a backend that serves all requests that are not matched with other backends. This allows the flexibility of adding new web applications without redefining existing backend URLs or adding new backends. Defining a default backend is done by configuring the URL path, "/", to the desired backend.

4.3 Bindings

Bindings provide a way to restrict the availability of backends to a subgroup of proxy servers. A binding is comprised of two elements: a proxy server and a backend.

While a backend has no associated binding, it is by default accessible from all proxy servers. As soon as a backend is associated with at least one binding, the backend becomes accessible only to the proxy servers defined in those bindings.

4.4 Load Balancing Algorithms

Several load balancing algorithms can be selected to affect the behavior of backend server selection.

4.4.1 Round-Robin

In the round-robin load balancing algorithm, each server of a backend is selected sequentially.

4.4.2 Least Connected

In the least-connection load-balancing algorithm, a count of currently active connections to each backend server is kept. The selected server is the server with the fewest number of active connections.

4.4.3 Weights

The load balancing algorithms take into account the weight defined for each backend server. Higher weight servers are chosen proportionally more often than lower weight servers. In the round-robin scheme, this (directly) means that servers are chosen more often. In the least-connected scheme, it means that servers are chosen to make the ratio of active connections equal to the ratio of weights.

4.5 Server Affinity

The Load Balancer supports server affinity. This allows clients to reliably reach the same backend server over multiple requests and connections.

4.5.1 Cookie

The Load Balancer can automatically include a cookie in HTTP responses to identify the backend server used. The client only needs to include this cookie in further requests to be routed to the same backend server.

The cookie server affinity scheme is useful when the load balanced application is stateful or makes use of sessions. In these cases, the client must consistently be able to reach the same backend server.

Applications that use sessions usually have some sort of authentication mechanism where users can enter their credentials. Without the cookie server affinity scheme, clients would constantly have to resend their credentials since the session established on a backend server is not valid on another backend server.

4.5.2 URL

In the URL server affinity scheme, the whole of the request's URL is hashed and used to locate the backend server in a hash table. Therefore, the same URL is always routed to the same backend server.

For example, the following two URLs would potentially reach different servers under this scheme:

http://broadsoft.com/webapp?user=1&group=2

http://broadsoft.com/webapp?group=2&user=1

4.5.3 URL Parameter

In the URL parameter server affinity scheme, the query string of the URL is parsed and the value of the configured parameter is extracted and hashed. This hash is used to locate the backend server in a hash table. Therefore, requests containing the same URL parameter are always routed to the same backend server. If the URL parameter is absent from the request, the round-robin algorithm is chosen instead.

For example, the following two URLs would reach the same backend server if the URL parameter scheme was set to "user":

http://broadsoft.com/webapp?user=1&group=2

http://broadsoft.com/webapp?group=2&user=1

4.6 Web Container

4.6.1 Legacy Server Affinity

Versions of the Xtended Services Platform prior to the enhancements described in this document applied a form of server affinity that is incompatible with the solution introduced by the Load Balancer application. When connecting to the Web Container of an Xtended Services Platform, the client would receive a response redirecting the next requests to the specific address of the Xtended Services Platform. Although this allows further requests to always reach the same Xtended Services Platform, it also causes clients to bypass the Load Balancer in subsequent requests.

It is therefore necessary to disable these redirection responses. This can be achieved by adding an alias to the list of HTTP aliases. When the Xtended Services Platform receives an HTTP request with the host header identical to one of its aliases, it does not perform redirection.

4.6.2 SSL Redirection

Some applications enforce rules regarding whether some pages require that SSL be enabled and some others require that SSL be disabled. These rules take the form of redirection responses. When the application is fronted by a reverse proxy such as a Load Balancer, it cannot infer whether the client used a secure connection to connect to the reverse proxy.

These redirection rules must be disabled to prevent redirection loops.

Redirection loops can happen when the Load Balancer acts as the SSL termination node. HTTP traffic is encrypted between the client and the Load Balancer but then is sent as clear text between the Load Balancer and the application. Since the application desires to have an encrypted connection, it sends a redirection response to the client. The client, on the other hand, already uses encryption on its connection, detects the redirection loop, and aborts the request.

4.6.2.1 First Solution: Full Dual Secure and Insecure Load Balancing

The first solution requires mapping unsecure connections from the client to unsecure connections on the server and secure connections from the client to secure connections on the server. This can be achieved by having both a secure and unsecure frontend bound to both a secure and unsecure backend containing the application. When the client receives an SSL redirection response, it simply connects to the other interface of the Load Balancer and the Load Balancer correspondingly uses the other backend.

This solution has the benefit of being transparent to the application. No additional configuration is required on its part. Moreover, the application behaves identically when a client is connected directly, without going through a Load Balancer.

4.6.2.2 Second Solution: Disabling SSL Redirection

The second solution requires disabling the application SSL redirections. The application should be configured to only accept one type of connection, secure or insecure. The Load Balancer should be similarly configured to connect to this secure or insecure interface. This way, the application does not attempt to send redirection responses to clients behind the Load Balancer.

This solution requires the ability to configure each application. CommPilot has a general configuration setting in the CLI, disabling SSL (off) or enabling it on all pages (full). It can be found in the CLI at the following location.

```
XSP_CLI/Applications/CommPilot/GeneralSettings/sslMode
```

4.6.3 Other Redirections

The CommPilot application also performs redirections when it receives requests for users that are not stored on a compatible Application Server, as reported by location requests to the Network Server. This can happen when the Xtended Services Platform farm is being upgraded and multiple release versions cohabit. In these situations, the application redirects the client to a compatible Xtended Services Platform.

This type of redirection cannot be prevented as it is necessary to process the request on the correct Xtended Services Platform. In this case, the redirect response is allowed to reach the client. The client then accesses the target Xtended Services Platform directly, thereby bypassing the Load Balancer.

4.7 Dynamic Configuration Reload

When the proxy servers, backends, or routing rule definitions are modified, an explicit reload of the configuration must be performed for the new changes to take effect. A special command is provided to perform this task.

The Load Balancer is by default configured to automatically reload its configuration automatically without requiring a restart of the application. With this feature enabled, modifying the Load Balancer configuration automatically triggers the Load Balancer to reconfigure itself. The configuration is gracefully reloaded after a period of time when the configuration is not modified. This allows the administrator to configure many parameters and only reload the configuration once. This grace period before a configuration reload can be configured.

When the configuration reload is issued, the following actions are performed:

1) A new Load Balancer instance is started and attempts to bind to its configured interfaces.

- If the bind succeeds, a signal is sent to the old instance to shut down. The Load Balancer application has successfully reloaded its configuration and is ready for processing.
- If the bind fails, a signal is sent to the old instance to temporarily stop listening on its sockets. The new instance attempts once again to bind to its interfaces. If it fails again, a signal is sent to the old instance telling it to resume listening. Otherwise, the bind is successful and the old instance is signaled to shut down. The Load Balancer application has successfully reloaded its configuration and is ready for processing.

2) When the old instance is shut down, connections that were already established before the configuration reload are allowed to finish their pending transactions before being disconnected. When the clients reestablish their connections, they connect to the new instance.

4.8 Health Checks

The Load Balancer monitors the health of every server defined in each backend. This is done by periodically attempting to connect to the target server. If this connection succeeds, the server is deemed operational and requests can be routed to this server. In the event of a failure to connect, the server is deemed not to be operational and no requests are routed to that server.

The Load Balancer continues to monitor the health of each server, regardless of its status, and if a server becomes operational again, requests resume being routed to that server.

4.8.1 Out-of-Band Health Checks

Out-of-band health checks are performed periodically. These checks consist of sending an HTTP request to each of the monitored backend servers. The HTTP response status code is then compared to a list of configured status codes to determine whether the health check is successful.

A certain number of consecutive failed health checks are required before a backend server is deemed offline. Similarly, a certain number of consecutive successful health checks are required before a backend server is deemed online. These numbers are configurable.

4.8.2 In-Band Health Checks

In-band health checks are performed on regular traffic passing through the Load Balancer. When an HTTP response is received from a backend server, its status code is processed. If the status code is less than 100 or greater than 499, with the exception of 502 and 507, the health check fails.

A certain number of consecutive failed in-band health checks are required before a backend server is deemed offline. This number is configurable. Since the in-band health checks cannot be used to monitor a failed backend server, the out-of-band health checks are required to restore a backend server to usual activity.

4.9 SSL Termination

The Load Balancer also acts as an SSL endpoint. This is necessary to read the HTTP request headers to apply the routing rules. As an SSL endpoint, the Load Balancer has its own list of certificates and enabled cipher suites. It also supports client authentication via the definition of trusts per interface.

Details of the SSL connection are inserted into the HTTP request forwarded to the backend server. This allows the backend server or web application to have access to the client's certificate details.

4.9.1 Client Authentication

Client authentication can be enabled on a per-server and per-backend basis. Both the server and the backend's client authentication settings must be enabled for a request going through that server and backend to be authenticated.

When client authentication is enabled on a proxy server, the client is asked for its certificate. If the client provides a certificate but the certificate is invalid, the connection fails immediately. If the client provides a valid certificate or if it does not provide a certificate, the request is allowed to be inspected further by the backend.

The backend is responsible for enforcing client authentication. If client authentication is enabled on the backend, the request must contain a valid certificate to be proxied further. If the request does not contain a certificate, a response with the error code 403 is returned.

Note that if the backend does not require client authentication but the client provides an invalid certificate anyway, the request fails. If the client had not included any certificate, then the request would have succeeded.

4.9.2 Web Container Configuration

4.9.2.1 Server Connections

The connections established by the Load Balancer to the backend servers can be configured to use the SSL. When the Load Balancer establishes a secure connection to a backend server, it does not validate the certificate provided by the remote host. Similarly, it does not present a certificate for the remote host to verify.

Therefore, the remote host's Web Container must be configured to not enforce client authentication on its interface. This does not prohibit the possibility of enabling client authentication on the Web Container on a per-web application basis.

4.9.2.2 Client Certificate Extraction

The Web Container can be configured to extract the client SSL information from the *ssl_client_cert, ssl_cipher, ssl_session_id,* and *ssl_cipher_usekeysize* HTTP headers. The *ssl_client_cert* header is automatically added by the Load Balancer when client authentication is enabled on both the proxy server and the backend server. To prevent any client from incorporating this header, the Web Container only extracts the client certificate from requests originating from a configurable list of IP addresses. Therefore, the Load Balancer's IP addresses must be added to that list when the Web Container is being load balanced and needs the client certificate information. Alternatively, the feature can be enabled, but with an empty list of IP addresses. In this case, no verification is made on the origin of the request and the headers are always extracted.

4.10 Connection Management

The Load Balancer uses sessions internally to keep track of incoming connections. Each client connection is associated with a session.

When a request is received, the Load Balancer chooses a backend server to handle the request according to its server affinity and load balancing rules. The connection that is established to this server is dedicated to that session and is not shared with other sessions.

When a subsequent request is received for the same session, the Load Balancer once again chooses the backend server. If the same backend server is chosen and the previous connection is still open, that connection is reused. Otherwise, the previous connection is closed and a new connection established.

As long as the requests associated with a session are being routed to the same backend server, the Load Balancer tries to reuse the backend connection as long as possible.

4.11 Application Layout

The Load Balancer application consists of two separate containers. The first container performs the load balancing and handles all connections. The second container is a helper controller that gathers the logs and statistics for the Load Balancer. The statistics are extracted by the controller from the Load Balancer using a local, dedicated HTTP interface. The controller then exposes these statistics to the platform Simple Network Management Protocol (SNMP) agent on behalf of the Load Balancer. The logs are pushed from the Load Balancer to a local UDP socket. The controller listens on that socket and receives the logs. They are then written to disk using the platform's logging infrastructure.

4.12 Multi-Process Operation

The Load Balancer supports a multi-process operation. In this mode, multiple Load Balancer processes are spawned at start-up. The different processes of the Load Balancer are independent and do not share information. When a new connection is established on the HTTP interface, one of the processes accepts the connection.

Since each process of the application generates its own statistics independently, the helper controller gathers the statistics of each process and provides a coalesced view of the Load Balancer. Since the helper controller must connect to each process separately, each Load Balancer process has its own dedicated HTTP interface for statistics monitoring.

Health checks are also performed by each Load Balancer process independently. It is therefore possible that the processes disagree on which backend servers are online or offline.

4.13 Logging

The Load Balancer produces both access logs and general health logs in the syslog format. The access logs are sent to the Load Balancer controller via a dedicated UDP port to be written to disk. On the other hand, the general logs are sent to the Log Server application via its UDP syslog port and are written to disk via the platform's logging infrastructure.

4.13.1 Log Server Improvements

A new listening interface is added to the Log Server application to support receiving syslog-formatted logs emitted by the Load Balancer. Logs received on this interface are automatically logged using the Load Balancer log configuration.

4.14 Server Name Indication Support

The Load Balancer supports Server Name Indication (SNI) and can present different certificates during the SSL handshake, in addition to having a global certificate per interface. Each additional certificate is bound to an alias and the aliases are themselves bound to one or many interfaces.

When an SSL handshake is received with an SNI extension, the Load Balancer first tries to find an alias that corresponds to the desired server name.

- If this alias exists and is bound to the current interface, then its associated certificate is presented.
- If this alias cannot be found or the SNI extension is absent, then the global interface certificate is presented.

When defining a new alias, a self-signed certificate is automatically created for this alias. An alias always has an associated certificate. To stop the Load Balancer from presenting an alias's certificate, simply delete the alias. The Load Balancer then returns to using the interface certificate.

The aliases and their certificates are managed through the CLI. For more information about configuring the aliases, see the *ProxyAlias* CLI context.

5 Service Interactions

5.1 Service Precedence

This is not applicable.

5.2 Service Interactions

5.2.1 Template for Specific Service/Policy Interaction

This is not applicable.

6 System Management

6.1 Accounting

This is not applicable.

6.2 Monitoring and Troubleshooting

The Load Balancer supports logging. Logs include both access logs and failed health checks and service restorations. The access logs are emitted from the Load Balancing main application to a controller process that runs alongside it through a UDP local interface. The helper process is used to gather these logs and write them using the platform's logging facility. The other logs are sent to the Log Server application for processing and they are also written to disk using the platform's logging facility.

6.2.1 Log Files Description

Both the Load Balancer and its controller application generate logs in their own files.

The Load Balancer populates these files with the information in the following table.

Log File Name	Contents
haproxycontainerpmd	Output of the Load Balancer process monitor.
haproxyOutput	Output of the Load Balancer, used during start-up. Errors preventing the Load Balancer from starting can be found here.
haproxy	Proxy servers and backends going in and out of service. These messages are emitted when a health check fails or succeeds and when the Load Balancer automatically reloads its configuration. Service interruptions are logged at the *Warn* severity. Service restorations are logged at the *Notice* severity.
access_log	Information about HTTP requests going through the Load Balancer.

The Load Balancer controller populates these files with the information in the following table.

Log File Name	Contents
haproxyControllercontainerpmd	Output of the Load Balancer controller process monitor.
haproxyControllerOutput	Output of the Load Balancer controller JVM. JVM garbage collection and heap information can be found here.
haproxyController	Information about automatic configuration reload, including Load Balancer start-up output. Errors preventing the Load Balancer from reloading its configuration can be found here. General information about subsystems and *stats collector* errors.

6.2.2 Alarms and PMs

The Load Balancer's counters can be accessed using a separate HTTP interface on a dedicated TCP port. The counters are provided individually for each proxy server, backend, and backend server.

In multi-process mode, each Load Balancer process must open a separate TCP port to access the processes' counters independently. A helper process is used to gather the counters, generate the appropriate statistics, and interface with the SNMP daemon.

The PMs available for each *proxy server* are the following:

- The number of current connections
- The maximum number of connections
- The cumulative number of connections
- The number of bytes received from the client
- The number of bytes sent to the client
- The number of request errors
- The number of new connections over last elapsed second
- The number of responses with 1xx code
- The number of responses with 2xx code
- The number of responses with 3xx code
- The number of responses with 4xx code
- The number of responses with 5xx code
- The number of responses with other codes
- The number of requests over last elapsed second
- The maximum rate of requests per second
- The cumulative number of requests

The PMs available for each *backend* are the following:

- The number of queued requests
- The maximum number of queued requests
- The number of current connections
- The maximum number of connections
- The cumulative number of connections
- The number of bytes received from the server
- The number of bytes sent to the server
- The number of connection errors for all servers of this backend
- The number of times a connection to a server was retried
- The number of times a request was redispatched to another server
- The number of times this backend became "inoperational" because all its servers were offline
- The number of seconds since the backend changed its operational state
- The total time the backend was "inoperational" (in seconds)
- The number of new connections over the last elapsed second
- The number of responses with 1xx code

- The number of responses with 2xx code
- The number of responses with 3xx code
- The number of responses with 4xx code
- The number of responses with 5xx code
- The number of responses with other codes
- The number of data transfers aborted by the client
- The number of data transfers aborted by the server
- The average time spent in a Load Balancer queue over the last 1024 requests
- The average time waiting to establish a connection to a backend server over the last 1024 requests
- The average time waiting for the response over the last 1024 requests

Many of the counters available for each backend are also available for each backend server. The PMs available for each server include:

- The number of queued requests
- The maximum number of queued requests
- The number of current connections
- The maximum number of connections
- The cumulative number of connections
- The number of bytes received from the server
- The number of bytes sent to the server
- The number of connection errors encountered with this server
- The number of times a connection to this server was retried
- The number of times a request sent to this server was redispatched to another server
- The number of failed health checks
- The number of times this server became offline
- The number of seconds since the backend changed operational state
- The total time this server was offline (in seconds)
- The number of times this server was selected by the load balancing algorithm
- The number of new connections during the last elapsed second
- The number of responses with 1xx code
- The number of responses with 2xx code
- The number of responses with 3xx code
- The number of responses with 4xx code
- The number of responses with 5xx code
- The number of responses with other codes
- The number of data transfers aborted by the client
- The number of data transfers aborted by the server

- The average time spent in a Load Balancer queue over the last 1024 requests
- The average time waiting to establish a connection to a backend server over the last 1024 requests
- The average time waiting for the response over the last 1024 requests

6.2.3 Commands and Utilities

This is not applicable.

6.3 Device Management

This is not applicable.

7 Restrictions and Limitations

7.1 Functional Limitations

There are no known functional limitations.

7.2 Upgrade/Rollback Limitations

There are no known upgrade or rollback limitations.

7.3 Enterprise Migration Restrictions

This is not applicable.

8 Service Patch Information

This feature is not being patched.

8.1 Functional Differences

8.1.1 Provisioning Differences

This is not applicable.

8.1.2 Feature Operational Differences

This is not applicable.

8.1.3 Service Interaction Differences

This is not applicable.

8.1.4 System Management Differences

This is not applicable.

8.2 Feature Activation Impacts

8.2.1 Method of Activation

This is not applicable.

8.2.2 Activatable Feature ID and Dependencies

This is not applicable.

9 Provisioning Interface Impacts

9.1 Centralized Configuration Data

The Load Balancing application is a new Cisco BroadWorks application that encapsulates the HAProxy third-party software. It runs on the Xtended Services Platform.

This feature introduces two new containers under the *LoadBalancer* application. The following table lists the new required components.

Application	Container	Subsystem	Reference
LoadBalancer	haproxy	*loadBalancerSubsystem*	Factory configuration settings defined in section *9.1.1.1 loadBalancerSubsystem Configuration.*
		logSubsystem	Existing. Factory configuration settings defined in section *9.1.1.3 logSubsystem.*
		logProfileSubsystem	Existing. Not configurable.
	haproxyController	*haproxyControllerSubsystem*	Factory configuration settings defined in section *9.1.2.1 loadBalancerControllerSubsystem Configuration.*
		logSubsystem	Existing. Factory configuration settings defined in section *9.1.2.2 logSubsystem.*
		logProfileSubsystem	Existing. Not configurable.
		jvmStatsCollector	Existing. Not configurable
		performance	Existing. Not configurable.
		bcct	Existing. Not configurable.
		snmp	Existing. Not configurable.

The *WebContainer* application is modified.

Application	Container	Subsystem	Reference
WebContainer	tomcat	*generalSettings*	Factory configuration settings defined in section *9.1.3.1 generalSettings.*
		clientAuthProxy	Factory configuration settings defined in section *9.1.3.2 clientAuthProxy Configuration.*

The *LogServer* application is modified. It is also now included in the Xtended Services Platform Server.

Application	Container	Subsystem	Reference
LogServer	LogServer	*logserverSubsystem*	Factory configuration settings defined in section *9.1.4.1 logserverSubsystem.*

9.1.1 haproxy Container

Note that a restart or soft configuration reload of the Load Balancer is required if any element in this section is modified.

9.1.1.1 loadBalancerSubsystem Configuration

Name	Type	Content Restrictions	Default Value	Description
generalSettings	Not applicable			Provides the ability to view and modify the general configuration settings.
backends	Collection			Provides the ability to view and modify the configuration settings for backends, which permit the definition of backend servers.
bindings	Collection			Provides the ability to view and modify the configuration settings for bindings, which restricts how backends are accessed from HTTP interfaces.

9.1.1.1.1 generalSettings

Name	Type	Content Restrictions	Default Value	Description
nbProc	Integer	1 through 64	1	This parameter specifies the number of processes to start.
statPortRangeStart	Integer	1025 through 65535, and *statPortRangeStart* + nbProc < 65536	9000	This parameter specifies the start of the local TCP port range used by the Load Balancer to expose its statistics interfaces. The range starts at this value and its size is equals the number of processes started.

9.1.1.1.2 automaticReload

Name	Type	Content Restrictions	Default Value	Description
enabled	Boolean		true	This parameter specifies the number of processes to start.
delay	Seconds	1 through 3600	30	This parameter specifies the delay after the last configuration change when the Load Balancer configuration is reloaded automatically.

9.1.1.1.3 backends

The key for this collection is the *backendName* attribute.

Name	Type	Content Restrictions	Default Value	Description
backendName	String	1 through 255 characters		This parameter specifies the name of this backend.
servers	Collection			Provides the ability to view and modify the configuration settings for backend servers.
urls	Collection			Provides the ability to view and modify the configuration settings for URLs, to determine which URL prefixes this backend server.
healthcheck	Not applicable			Provides the ability to view and modify the configuration settings for health checks.
loadBalancingAlgorithm	Enum	leastConn, roundRobin		This parameter specifies the load balancing algorithm used for this backend. The *leastConn* algorithm dynamically selects the server with the least number of active connections, while the *roundRobin* algorithm selects the servers in a static order.
serverAffinity	Enum	url, urlParam, cookie, disabled		This parameter specifies the type of server affinity scheme used for this backend to permit clients of the Load Balancer to be consistently served by the same backend server. ▪ The “url” scheme uses the whole URL as the key. ▪ The “urlParam” scheme uses a named URL parameter as the key. ▪ The “cookie” scheme uses a named cookie as the key. The “disabled” scheme disables this feature altogether.
urlParam	String	1 through 255 characters		This parameter specifies the name of the URL parameter to use in conjunction with the URL parameter server affinity scheme.
cookieName	String	1 through 255 characters		This parameter specifies the name of the cookie to use in conjunction with the cookie server affinity scheme.

9.1.1.1.3.1 servers

The keys for this collection are the *host* and *port* attributes.

Name	Type	Content Restrictions	Default Value	Description
serverName	String	1 through 255 characters		This parameter specifies the name of this server entry which is used for identification in server affinity schemes.

Name	Type	Content Restrictions	Default Value	Description
host	NetAddress			This parameter specifies the host name or IP address of the backend server.
port	Integer	1 through 65535		This parameter specifies the port of the backend server.
secure	Boolean			This parameter controls whether connections to this server uses the Secure Sockets Layer (SSL).
weight	Integer	0 through 256	10	This parameter specifies the relative bias toward this server by the load balancing algorithm.
enabled	Boolean		true	This parameter controls whether this server is enabled or disabled. Disabled servers are never used by the Load Balancer.

9.1.1.1.3.2 urls

The key for this collection is the *url* attribute.

Name	Type	Content Restrictions	Default Value	Description
url	URI	1 through 255 characters		This parameter specifies the URL prefix that is served by this backend.

9.1.1.1.3.3 healthcheck

Name	Type	Content Restrictions	Default Value	Description
inbandFall	Integer	0 through 100	2	This parameter specifies the number of consecutive in-band failed health checks that put the associated server offline.
outofbandRise	Integer	0 through 100	3	This parameter specifies the number of consecutive out-of-band successful health checks that put the associated server online.
outofbandFall	Integer	0 through 100	2	This parameter specifies the number of consecutive out-of-band failed health checks that put the associated server offline.
outofbandMethod	String	1 through 255 characters	HEAD	This parameter specifies the HTTP method used by the out-of-band health check to verify the state of the backend server.
outofbandUrl	URI	1 through 255 characters	/	This parameter specifies the URL used by the out-of-band health check to verify the state of the backend server.
outofbandInterval	Seconds	1 through 300	2	This parameter specifies the interval of time in seconds between two out-of-band health checks.

Name	Type	Content Restrictions	Default Value	Description
outofbandTimeout	Seconds	1 through 300	2	This parameter specifies the maximum response time in seconds for the server to respond to a health check request.

9.1.1.1.4 bindings

The keys for this collection are the *interfaceIp, port*, and *backend* attributes.

Name	Type	Content Restrictions	Default Value	Description
interfaceIp	IPAddress			This parameter specifies the local IP address that makes up this binding.
port	Integer	1 through 65535		This parameter specifies the local port that makes up this binding.
backend	String	1 through 255 characters		This parameter specifies the backend name that makes up this binding.

9.1.1.2 Interfaces Configuration

9.1.1.2.1 http

Name	Type	Content Restrictions	Default Value	Description
maxConnections	Integer		10000	This parameter specifies the maximum number of incoming connections.
maxQueuedConnections	Integer		128	This parameter specifies the size of the TCP backlog for each listening socket. The operating system uses this parameter as a hint.
connectTimeout	Seconds	1 through 300	5	This parameter specifies the number of seconds the Load Balancer waits for a connection to a backend to be established.
keepAliveTimeout	Seconds	1 through 300	5	This parameter specifies the number of seconds of inactivity on a connection after which it is closed by the Load Balancer.
queueTimeout	Seconds	1 through 300	5	This parameter specifies the number of seconds a request waits in the Load Balancer if it cannot be proxied immediately.

Name	Type	Content Restrictions	Default Value	Description
ciphers	String	1 through 512 characters	ECDHE-RSA-AES256-SHA384:AES256-SHA256:RC4:HIGH:!MD5:!aNULL:!EDH:!AESGCM	This parameter specifies the enabled cipher suites for all servers using the OpenSSL syntax. These ciphers are used when per-server ciphers are not defined.

9.1.1.2.1.1 servers

The keys for this collection are the *interfaceIp* and *port* attributes.

Name	Type	Content Restrictions	Default Value	Description
interfaceIp	IPAddress			This parameter specifies the interface IP address.
port	Integer	1 through 65535		This parameter specifies the port number.
name	String	1 through 255 characters		This parameter specifies the name of the proxy server.
secure	Boolean			This parameter controls server security.
clientAuthReq	Boolean			This parameter determines whether the server requests the client identity. ▪ When the value is set to "true", the server requests the client identity. ▪ When the value is set to "false", the server does not request the client identity. When a client receives a request for identity, it sends a certificate.
clusterFqdn	String	1 through 80 characters		This parameter specifies the fully qualified domain name (FQDN) of the cluster (for example, *xsp1.mtl.broadsoft.com*).
ciphers	String	1 through 512 characters		This parameter specifies the cipher suites enabled for secure connections to this server. The cipher suites are defined using the OpenSSL syntax.

9.1.1.3 logSubsystem

The standard log subsystem is used. The factory settings are specified in the following subsections.

9.1.1.3.1 generalSettings

Name	Value
enabled	true

Name	Value
priority	4
maxQueueSize	50000
severity	Info
showThreadName	false

9.1.1.3.2 inputChannels

Name	Enabled	Severity	Categories
Generic	true	nil	

9.1.1.3.3 outputChannels

Name	Enabled	Directory	FilePrefix	FileSizeMegs	NumberOfFiles
File	true	*/var/broadworks/logs/loadbalancer*	Haproxy	30	200

9.1.1.4 accessLogs Subsystem

The standard *accessLogs* subsystem is used. The factory settings are specified in the following subsections.

9.1.1.4.1 generalSettings

Name	Value
enabled	true
priority	4
maxQueueSize	50000
severity	Info
showThreadName	true

9.1.1.4.2 outputChannels

Name	Enabled	Directory	FilePrefix	FileSizeMegs	NumberOfFiles
File	true	*/var/broadworks/logs/loadbalancer*	access_log	8	600

9.1.2 haproxyController Container

9.1.2.1 loadBalancerControllerSubsystem Configuration

Name	Type	Content Restrictions	Default Value	Description
generalSettings	Not applicable			Provides the ability to view and modify the general configuration settings.

9.1.2.1.1 generalSettings

Changing any of the values in this table requires restarting the Load Balancer application.

Name	Type	Content Restrictions	Default Value	Description
logPort	Integer	1025 through 65535	8999	This parameter specifies the local UDP port used by the Load Balancer controller to receive the Load Balancer access logs.

9.1.2.2 logSubsystem

The standard log subsystem is used. The factory settings are specified in the following subsections.

9.1.2.2.1 generalSettings

Name	Value
enabled	true
priority	4
maxQueueSize	50000
severity	Info
showThreadName	false

9.1.2.2.2 inputChannels

Name	Enabled	Severity	Categories
Generic	true	nil	
BroadsoftCommonCommunicationTransport	true	Info	
BroadsoftCommonCommunicationTransportKeepAlive	true	Notice	
ServiceOS	true	nil	
SMAP	false	nil	

Name	Enabled	Severity	Categories
NameService	true	nil	
GcLog	true	Info	

9.1.2.2.3 outputChannels

Name	Enabled	Directory	FilePrefix	FileSizeMegs	NumberOfFiles
File	true	*/var/broadworks/logs/loadbalancer*	HaproxyController	30	200

9.1.3 tomcat Container

9.1.3.1 generalSettings

Name	Type	Content Restrictions	Default Value	Description
jvmRoute	String	1 through 255 characters	nil	This parameter specifies the name of this server instance that is appended to the JSESSIONID in HTTP responses.

9.1.3.2 clientAuthProxy Configuration

Name	Type	Content Restrictions	Default Value	Description
enable	Boolean		false	This parameter controls whether the client certificate information located in the HTTP headers is parsed and forwarded to the web applications. Enabling this option is useful when the client SSL connection is terminated by an external component.
networkAccessList	Collection			Provides the ability to view and modify the network access list settings.

9.1.3.2.1 networkAccessList

Name	Type	Content Restrictions	Default Value	Description
address	IPAddress			This parameter specifies the Internet Protocol (IP) address of the network element that sends client authentication information to the web container.
description	String	1 through 80 characters	nil	This parameter specifies an optional description for the network element.

9.1.4 LogServer Container

9.1.4.1 logserverSubsystem

9.1.4.1.1 generalSettings

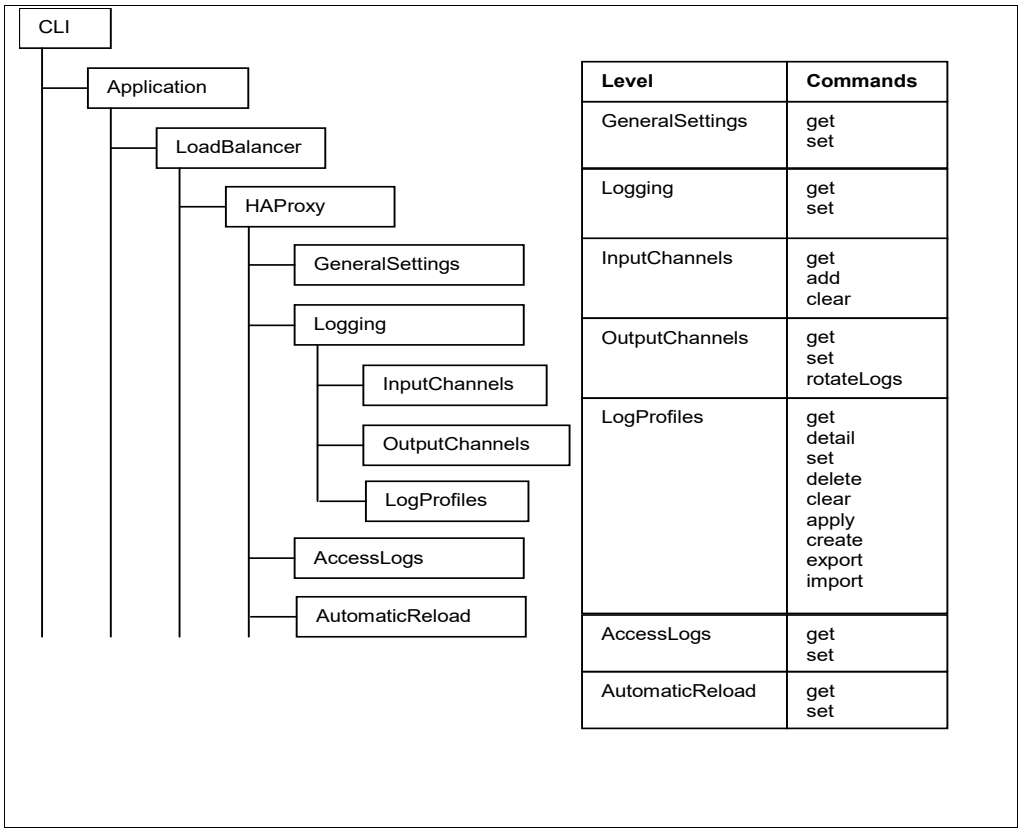
A new element is added to the LogServer container general settings.

Name	Type	Content Restrictions	Default Value	Description
syslogUdpPort	TcpUdpPort		4445	This parameter specifies the syslog server port.

9.2 CLI Impacts

9.2.1 Summary

A new application is added to the Xtended Services Platform CLI.



Level	Commands
GeneralSettings	get set
Logging	get set
InputChannels	get add clear
OutputChannels	get set rotateLogs
LogProfiles	get detail set delete clear apply create export import
AccessLogs	get set
AutomaticReload	get set

Figure 2 CLI Hierarchy Changes

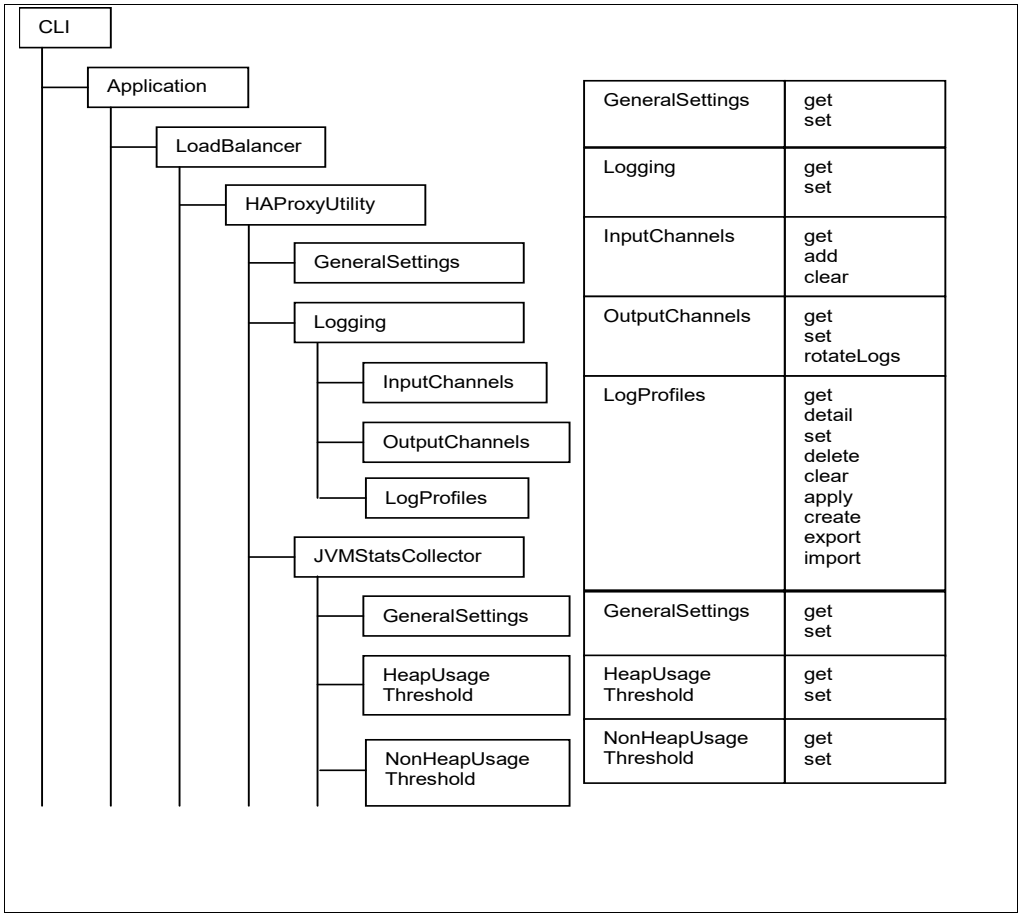
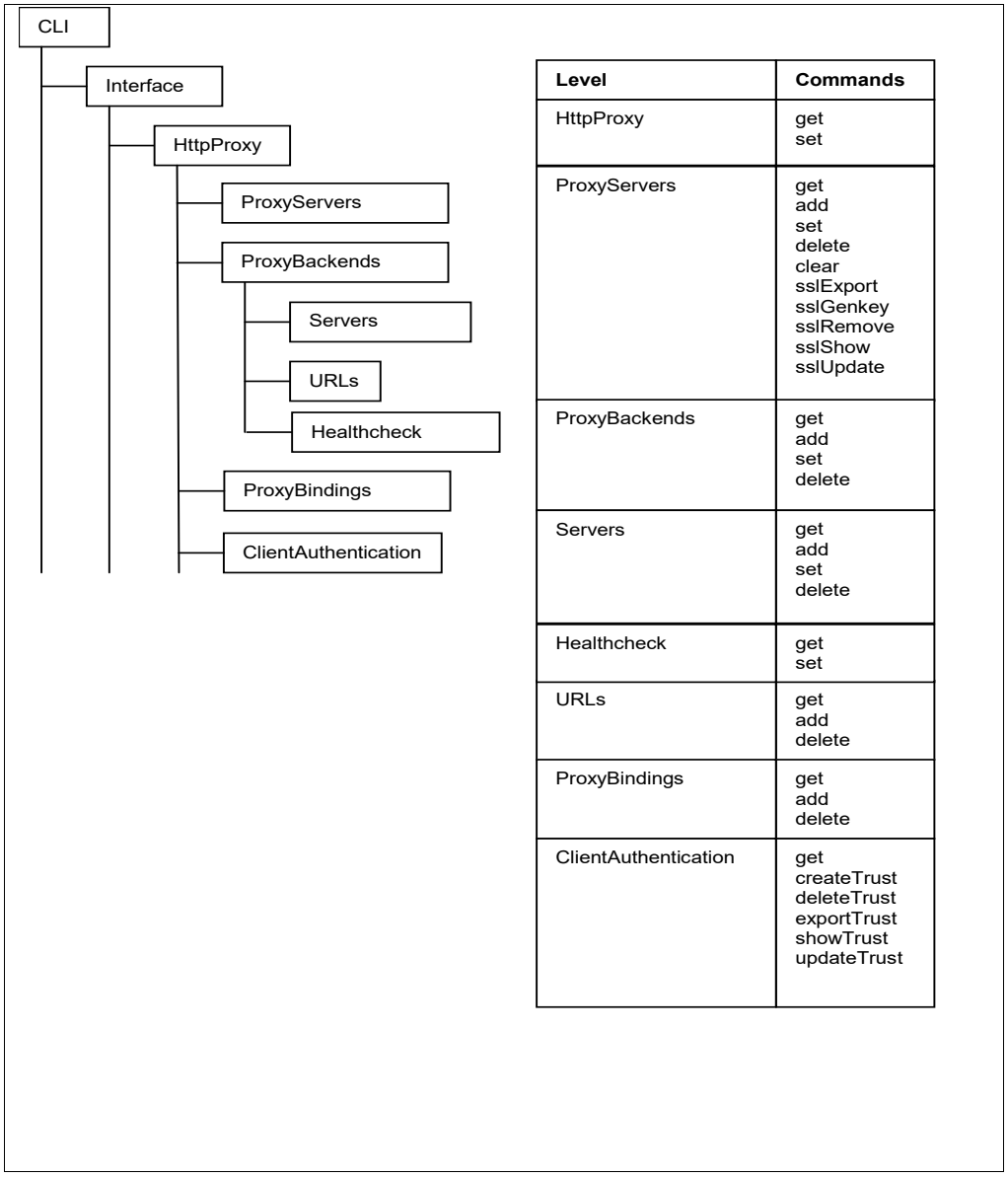


Figure 3 CLI Hierarchy Changes

A new interface is added to the Xtended Services Platform CLI.



Level	Commands
HttpProxy	get set
ProxyServers	get add set delete clear sslExport sslGenkey sslRemove sslShow sslUpdate
ProxyBackends	get add set delete
Servers	get add set delete
Healthcheck	get set
URLs	get add delete
ProxyBindings	get add delete
ClientAuthentication	get createTrust deleteTrust exportTrust showTrust updateTrust

Figure 4 CLI Hierarchy Changes

The Web Container application is modified to add support for third-party load balancing and client certificate forwarding.

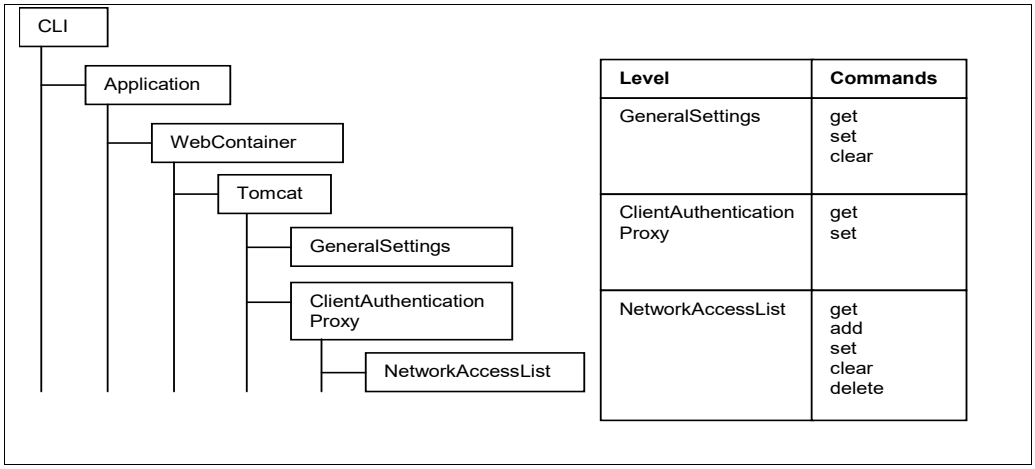


Figure 5 CLI Hierarchy Changes

The Log Server application is added to the Xtended Services Platform and modified to add support for a syslog UDP port.

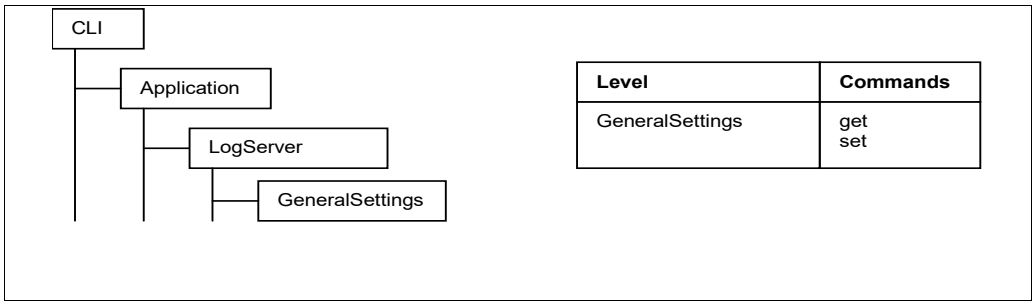


Figure 6 CLI Hierarchy Changes

9.2.2 Interface/HttpProxy

1) Interface/HttpProxy/get

Location within CLI Tree

```
CLI/Interface/HttpProxy
```

Command Format

```
get          get takes no parameter
```

Command Definition and Usage

```
This command is used to view the HTTP proxy general settings.
```

Example

```
$ CLI/Interface/HttpProxy/ProxyServers> get

  keepAliveTimeoutInSeconds = 5
  clientTimeoutInSeconds = 300
  serverTimeoutInSeconds = 300
  tunnelTimeoutInSeconds = 300
  connectionTimeoutInSeconds = 40
  queueTimeoutInSeconds = 5
  maxConnections = 10000
  maxQueuedConnection = 100
  ciphers = ECDHE-RSA-AES256-SHA384:AES256-
SHA256:RC4:HIGH:!MD5:!aNULL:!EDH:!AESGCM
```

2) Interface/HttpProxy/set

Location within CLI Tree

```
CLI/Interface/HttpProxy
```

Command Format

```
set
                    <attribute>, Multiple Choice =
                    {keepAliveTimeoutInSeconds, clientTimeoutInSeconds,
                    serverTimeoutInSeconds, tunnelTimeoutInSeconds,
                    connectionTimeoutInSeconds, queueTimeoutInSeconds,
                    maxConnections, maxQueuedConnection, ciphers}
                        <keepAliveTimeoutInSeconds>, Integer {1 to 300}
                        <clientTimeoutInSeconds>, Integer {1 to 300}
                        <serverTimeoutInSeconds>, Integer {1 to 300}
                        <tunnelTimeoutInSeconds>, Integer {1 to 3600}
                        <connectionTimeoutInSeconds>, Integer {1 to 300}
                        <queueTimeoutInSeconds>, Integer {1 to 300}
                        <maxConnections>, Integer {1 to 2147483647}
                        <maxQueuedConnection>, Integer {1 to 2147483647}
                        <ciphers>, String {1 to 512 characters}
```

Command Definition and Usage

```
This command is used to modify the HTTP proxy general
settings.
```

Parameter Definitions

maxConnections	This parameter specifies the maximum number of incoming connections that the load balancer accepts.
maxQueuedConnections	This parameter specifies the size of the TCP listen backlog. When this backlog is full, further connections are refused.

connectionTimeoutInSeconds	This parameter specifies the amount of time the load balancer waits for a connection to a backend server to be established.
keepAliveTimeoutInSeconds	This parameter specifies the inactivity period of a session after which both its client and server connections are closed.
queueTimeoutInSeconds	This parameter specifies the amount of time a request waits in the load balancer if it is not proxied immediately.
ciphers	This parameter specifies the enabled cipher suites for all servers using the OpenSSL syntax. These ciphers are used whenever per-server ciphers are not defined.
clientTimeoutInSeconds	This parameter specifies the number of seconds of client inactivity on a connection after which it is closed by the load balancer.
serverTimeoutInSeconds	This parameter specifies the number of seconds of server inactivity on a connection after which it is closed by the load balancer.
tunnelTimeoutInSeconds	This parameter specifies the number of seconds of inactivity on a connection that was upgraded to a tunnel, such as a WebSocket connection, after which it is closed by the load balancer.

Example

```
$ CLI/Interface/HttpProxy> set ciphers ALL
```

9.2.3 Interface/HttpProxy/ProxyServers

1) Interface/HttpProxy/ProxyServers/get

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyServers
```

Command Format

```
get          get takes no parameter
```

Command Definition and Usage

```
This command is used to view the HTTP Proxy's Proxy Servers.
```

Example

```
$ CLI/Interface/HttpProxy/ProxyServers> get
  Interface     Port       Name  Secure  Client Auth Req  Cluster
Fqdn  Ciphers
=================================================================
=============
       1.2.3.4   443    1.2.3.4    true            false
         ALL
```

2) Interface/HttpProxy/ProxyServers/add

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyServers
```

Command Format

```
add              <interfaceIp>, Choice = {<IP addresses>}
                 <port>, Integer {0 to 65535}
                 <name>, IP address | host | domain (1 to 80 chars)
                 <secure>, Choice = {false, true}
                 <clientAuthReq>, Choice = {false, true}
                 <ciphers>, String {1 to 512 characters}
                 [<attribute>, Multiple Choice = {clusterFqdn}]
                   <clusterFqdn>, String {1 to 80 characters}
```

Command Definition and Usage

```
This command is used to add a Proxy Server.
```

Parameter Definitions

interface	This parameter specifies the interface IP address.
port	This parameter specifies the port number.
name	This parameter specifies the name of the proxy server.
secure	This parameter controls server security.
clientAuthReq	This parameter determines whether the server requests the client identity. When the value is set to "true", the server requests the client identity. When the value is set to "false", the server does not request the client identity. When a client receives a request for identity, it sends a certificate.
ciphers	This parameter specifies the enabled cipher suites for this server using the OpenSSL syntax.
attribute	The name of attributes to include through the add command.
clusterFqdn	This parameter specifies the FQDN of the cluster (for example, xsp1.mtl.broadsoft.com).

Example

```
$ CLI/Interface/HttpProxy/ProxyServers> add 1.2.3.4 443
1.2.3.4 true false ALL
```

3) Interface/HttpProxy/ProxyServers/set

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyServers
```

Command Format

```
set             <interfaceIp>, Choice = {<IP addresses>}
                <port>, Integer {0 to 65535}
                <attribute>, Multiple Choice = {name, secure, clientAuthReq,
                clusterFqdn, ciphers}
                  <name>, IP address | host | domain (1 to 80 chars)
                  <secure>, Choice = {false, true}
                  <clientAuthReq>, Choice = {false, true}
                  <ciphers>, String {1 to 512 characters}
                  <clusterFqdn>, String {1 to 80 characters}
```

Command Definition and Usage

```
This command is used to modify a proxy server's settings.
```

Parameter Definitions

interface	This parameter specifies the interface Internet Protocol (IP) address.
port	This parameter specifies the port number.
attribute	The name of an attribute to modify.
name	This parameter specifies the name of the proxy server.
secure	This parameter controls server security.
clientAuthReq	This parameter determines whether the server requests the client identity. When the value is set to "true", the server requests the client identity. When the value is set to "false", the server does not request the client identity. When a client receives a request for identity, it sends a certificate.
ciphers	This parameter specifies the enabled cipher suites for this server using the OpenSSL syntax.
clusterFqdn	This parameter specifies the FQDN of the cluster (for example, xsp1.mtl.broadsoft.com).

Example

```
$ CLI/Interface/HttpProxy/ProxyServers> set 1.2.3.4 443
ciphers HIGH
```

4) Interface/HttpProxy/ProxyServers/delete

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyServers
```

Command Format

```
delete          <interfaceIp>, Choice = {<IP addresses>}
                <port>, Integer {0 to 65535}
```

Command Definition and Usage

```
This command is used to delete a proxy server.
```

Parameter Definitions

interface	This parameter specifies the interface IP address.
port	This parameter specifies the port number.

Example

```
$ CLI/Interface/HttpProxy/ProxyServers> delete 1.2.3.4 443
```

5) Interface/HttpProxy/ProxyServers/clear

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyServers
```

Command Format

```
clear           <interfaceIp>, Choice = {<IP addresses>}
                <port>, Integer {0 to 65535}
                <attribute>, Multiple Choice = {clusterFqdn}
```

Command Definition and Usage

```
This command is used to clear a proxy server's attributes.
```

Parameter Definitions (include default value, if any – only for Network Server CLI)

interface	This parameter specifies the interface IP address.

port	This parameter specifies the port number.
attribute	The name of an attribute to clear.

Example

```
$ CLI/Interface/HttpProxy/ProxyServers> clear 1.2.3.4 443
clusterFqdn
```

6) Interface/HttpProxy/ProxyServers/sslExport

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyServers
```

Command Format

```
sslExport       <interfaceIp>, Choice = {<IP addresses>}
                <port>, Integer {0 to 65535}
                <attribute>, Multiple Choice = {certificateFile, keyFile,
                chainFile}
```

Command Definition and Usage

This command is used to export Secure Sockets Layer (SSL) files (certificate, key, or chain file). The SSL file is taken from its location and copied to the */var/broadworks/tmp/* directory.

Parameter Definitions

interface	This parameter specifies the interface IP address.
port	This parameter specifies the port number.
attribute	The name of the file to export.

Example

```
$ CLI/Interface/HttpProxy/ProxyServers> sslExport 1.2.3.4 443
certificateFile
```

7) Interface/HttpProxy/ProxyServers/sslGenkey

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyServers
```

Command Format

sslGenkey	<interfaceIp>, Choice = {<IP addresses>} <port>, Integer {0 to 65535} [<attribute>, Multiple Choice = {keyLength, digest, country, stateOrProvince, city, organisationName, organisationUnit, emailAddress, challengePassword, commonName}] <keyLength>, Integer {512 to 16384} <digest>, Choice = {sha1, sha256} <country>, String {2 characters} <stateOrProvince>, String {1 to 50 characters} <city>, String {1 to 50 characters} <organisationName>, String {1 to 50 characters} <organisationUnit>, String {1 to 50 characters} <emailAddress>, String {1 to 50 characters} <challengePassword>, String {1 to 40 characters} <commonName>, String {1 to 255 characters}

Command Definition and Usage

This command is used to generate a Secure Sockets Layer (SSL) key for a secure proxy server. A generated Certificate Signing Request (*.csr*) file is written to the */tmp* directory under the *name.csr*, where name is the interface IP address.

Parameter Definitions

interface	This parameter specifies the interface IP address.
port	This parameter specifies the port number.
attribute	The name of the attribute to set.
keyLength	This parameter specifies the SSL key length or key size. The default value (when not specified) is "1024".
digest	This parameter specifies the message digest algorithm. The default value (when not specified) is "sha256".
country	This parameter specifies the country code international abbreviation code.
stateOrProvince	This parameter specifies the state or province.
city	This parameter specifies the name of the city.
organisationName	This parameter specifies the organization name.
organisationUnit	This parameter specifies the organization unit.
emailAddress	This parameter specifies the e-mail address.
challengePassword	This parameter specifies the challenge password.

commonName	This parameter specifies the common name. It must correspond to the host name of this proxy server. A "*" wildcard character may be present in the left-most name component of the host if the certificate is used on multiple servers having the same domain. For example, the common name for host xsp1.broadsoft.com could be *.broadsoft.com or xsp*.broadsoft.com.

Example

```
$ CLI/Interface/HttpProxy/ProxyServers> sslGenkey 1.2.3.4
443 digest sha256
```

8) Interface/HttpProxy/ProxyServers/sslRemove

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyServers
```

Command Format

sslRemove	<interfaceIp>, Choice = {<IP addresses>} <port>, Integer {0 to 65535} <attribute>, Multiple Choice = {chainFile}

Command Definition and Usage

This command is used to remove the Secure Sockets Layer (SSL) file (chain file).

Parameter Definitions

interface	This parameter specifies the interface IP address.
port	This parameter specifies the port number.
attribute	The type of file to remove.

Example

```
$ CLI/Interface/HttpProxy/ProxyServers> sslRemove 1.2.3.4
443 chainFile
```

9) Interface/HttpProxy/ProxyServers/sslShow

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyServers
```

Command Format

sslShow	<interfaceIp>, Choice = {<IP addresses>} <port>, Integer {0 to 65535}

Command Definition and Usage

This command is used to view the details of the X.509 certificate and certificate chain associated with the provided interface. This command reflects the information sent by the server to the client software.

Parameter Definitions

interface	This parameter specifies the interface IP address.
port	This parameter specifies the port number.

Example

```
$ CLI/Interface/HttpProxy/ProxyServers> sslShow 1.2.3.4
443
```

10) Interface/HttpProxy/ProxyServers/sslUpdate

Location within CLI Tree

CLI/Interface/HttpProxy/ProxyServers

Command Format

sslUpdate	<interfaceIp>, Choice = {<IP addresses>} <port>, Integer {0 to 65535} <attribute>, Multiple Choice = {certificateFile, keyFile, chainFile} <certificateFile>, String {1 to 255 characters} <keyFile>, String {1 to 255 characters} <chainFile>, String {1 to 255 characters}

Command Definition and Usage

This command is used to load and update a Secure Sockets Layer (SSL) certificate file. The certificate is taken from the specified path and copied to the proper location.

Parameter Definitions

interface	This parameter specifies the interface IP address.
port	This parameter specifies the port number.
attribute	The name of a Secure Sockets Layer (SSL) configuration attribute to modify.

certificateFile	This parameter specifies the file name of the certificate file.
keyFile	This parameter specifies the file name of the key file.
chainFile	This parameter specifies the file name of the certificate chain file.

Example

```
$ CLI/Interface/HttpProxy/ProxyServers> sslUpdate 1.2.3.4
443 certificateFile /path/to/certificate.cert
```

9.2.4 Interface/HttpProxy/ProxyBackends

1) Interface/HttpProxy/ProxyBackends/get

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyBackends
```

Command Format

```
get      get takes no parameter
```

Command Definition and Usage

```
This command is used to view the HTTP proxy backends.
```

Parameter Definitions

```
get     get takes no parameter
```

Example

```
$ CLI/Interface/HttpProxy/ProxyBackends> get
  Backend Name  Load Balancing Algorithm  Server Affinity  URL Param
Enable Cookie Persistency  Cookie Name
=============================================================================
=======================================
      backend1                leastConn         disabled      param
                    true        route
```

2) Interface/HttpProxy/ProxyBackends/add

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyBackends
```

Command Format

```
add             <backendName>, String {0 to 255 characters}
                <loadBalancingAlgorithm>, Choice = {leastConn, roundRobin}
                <serverAffinity>, Choice = {url, urlParam, cookie, disabled}
                <urlParam>, String {0 to 255 characters}
                <cookieName>, String {0 to 255 characters}
```

Command Definition and Usage

This command is used to add a backend to the HTTP proxy.

Parameter Definitions

backendName	This parameter specifies the name of the backend.
load Balancing Algorithm	This parameter specifies the type of load balancing algorithm to use.
server Affinity	This parameter specifies the type of server affinity scheme to use. This feature can also be disabled.
urlParam	This parameter specifies the name of the URL parameter to use when the urlParam server affinity scheme is enabled.
cookieName	This parameter specifies the name of the cookie to use when the cookie server affinity scheme is enabled.

Example

```
$ CLI/Interface/HttpProxy/ProxyBackends> add backend1 leastConn
cookie cookieName serverRoute
```

3) Interface/HttpProxy/ProxyBackends/set

Location within CLI Tree

CLI/Interface/HttpProxy/ProxyBackends

Command Format

```
set             <backendName>, String {0 to 255 characters}
                <attribute>, Multiple Choice = {loadBalancingAlgorithm,
                sessionAffinity, urlParam, cookieName}
                <loadBalancingAlgorithm>, Choice = {leastConn, roundRobin}
                <serverAffinity>, Choice = {url, urlParam, cookie, disabled}
                <urlParam>, String {0 to 255 characters}
                <cookieName>, String {0 to 255 characters}
```

Command Definition and Usage

This command is used to modify the settings of an HTTP proxy backend.

Parameter Definitions

backendName	This parameter specifies the name of the backend.
attribute	This parameter specifies the name of the attribute to modify.
load Balancing Algorithm	This parameter specifies the type of load balancing algorithm to use.
server Affinity	This parameter specifies the type of server affinity scheme to use. This feature can also be disabled.
urlParam	This parameter specifies the name of the URL parameter to use when the urlParam server affinity scheme is enabled.
cookieName	This parameter specifies the name of the cookie to use when the cookie server affinity scheme is enabled.

Example

```
$ CLI/Interface/HttpProxy/ProxyBackends> set backend1
loadBalancingAlgorithm roundRobin
```

4) Interface/HttpProxy/ProxyBackends/delete

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyBackends
```

Command Format

```
delete          <backendName>, String {0 to 255 characters}
```

Command Definition and Usage

```
This command is used to delete an HTTP proxy backend.
```

Parameter Definitions

backendName	This parameter specifies the name of the backend.

Example

```
$ CLI/Interface/HttpProxy/ProxyBackends> delete backend1
```

9.2.5 Interface/HttpProxy/ProxyBackends/Servers

1) Interface/HttpProxy/ProxyBackends/Servers/get

Location within CLI Tree

CLI/Interface/HttpProxy/ProxyBackends/Servers

Command Format

```
get             <backendName>, String {0 to 255 characters}
```

Command Definition and Usage

This command is used to view the HTTP proxy backend servers.

Parameter Definitions

backendName	This parameter specifies the name of the backend.

Example

```
$ CLI/Interface/HttpProxy/ProxyBackends/Servers> get backend1
        Host   Port   Server Name   Secure   Enabled
================================================
     1.2.3.4    80      example    false     true
```

2) Interface/HttpProxy/ProxyBackends/Servers/add

Location within CLI Tree

CLI/Interface/HttpProxy/ProxyBackends/Servers

Command Format

```
add             <backendName>, String {0 to 255 characters}
                <host>, String {0 to 255 characters}
                <port>, Integer {0 to 65535}
                <serverName>, String {0 to 255 characters}
                <secure>, Choice = {false, true}
                <weight>, Integer {0 to 256}
                <enabled>, Choice = {false, true}
```

Command Definition and Usage

This command is used to add a server to an HTTP proxy backend.

Parameter Definitions

backendName	This parameter specifies the name of the backend.
host	This parameter specifies the hostname of this server.
port	This parameter specifies the port number of this server.

serverName	This parameter specifies this server's unique name used for cookie server affinity. This name is exported in HTTP headers.
secure	This parameter controls server security.
weight	This parameter specifies the relative bias towards this server by the load balancing algorithm.
enabled	This parameter determines whether this server is enabled or disabled.

Example

```
$ CLI/Interface/HttpProxy/ProxyBackends/Servers> add backend1
4.3.2.1 443 server1 true 10 true
```

3) Interface/HttpProxy/ProxyBackends/Servers/set

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyBackends/Servers
```

Command Format

```
set             <backendName>, String {0 to 255 characters}
                <host>, String {0 to 255 characters}
                <port>, Integer {0 to 65535}
                <attribute>, Multiple Choice = {serverName, secure, enabled}
                    <serverName>, String {0 to 255 characters}
                    <secure>, Choice = {false, true}
                    <weight>, Integer {0 to 256}
                    <enabled>, Choice = {false, true}
```

Command Definition and Usage

```
This command is used to modify  an HTTP proxy backend server
properties.
```

Parameter Definitions

backendName	This parameter specifies the name of the backend.
host	This parameter specifies the hostname of this server.
port	This parameter specifies the port number of this server.
attribute	This parameter specifies the name of the attribute to modify.
serverName	This parameter specifies this server's unique name used for cookie server affinity. This name is exported in HTTP headers.
secure	This parameter controls server security.

weight	This parameter specifies the relative bias towards this server by the load balancing algorithm.
enabled	This parameter determines whether this server is enabled or disabled.

Example

```
$ CLI/Interface/HttpProxy/ProxyBackends/Servers> set backend1
4.3.2.1 443 enabled false
```

4) Interface/HttpProxy/ProxyBackends/Servers/delete

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyBackends/Servers
```

Command Format

```
delete          <backendName>, String {0 to 255 characters}
                <host>, String {0 to 255 characters}
                <port>, Integer {0 to 65535}
```

Command Definition and Usage

```
This command is used to remove an HTTP proxy backend server.
```

Parameter Definitions

backendName	This parameter specifies the name of the backend.
host	This parameter specifies the hostname of this server.
port	This parameter specifies the port number of this server.

Example

```
$ CLI/Interface/HttpProxy/ProxyBackends/Servers> delete backend1
4.3.2.1 443
```

9.2.6 Interface/HttpProxy/ProxyBackends/URLs

1) Interface/HttpProxy/ProxyBackends/URLs/get

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyBackends/URLs
```

Command Format

```
get             <backendName>, String {0 to 255 characters}
```

Command Definition and Usage

This command is used to view the URLs associated with the HTTP proxy backend.

Parameter Definitions

backendName	This parameter specifies the name of the backend.

Example

```
$ CLI/Interface/HttpProxy/ProxyServers/URLs> get backend1
                  URL
====================
    /path/to/webapp1
    /path/to/webapp2
```

2) Interface/HttpProxy/ProxyBackends/URLs/add

Location within CLI Tree

CLI/Interface/HttpProxy/ProxyBackends/URLs

Command Format

```
add             <backendName>, String {0 to 255 characters}
                <url>, String {0 to 255 characters}
```

Command Definition and Usage

This command is used to add a URL to an HTTP proxy backend.

Parameter Definitions

backendName	This parameter specifies the name of the backend.
url	This parameter specifies the URL prefix.

Example

```
$ CLI/Interface/HttpProxy/ProxyBackends/URLs> add backend1
/path/to/webapp3
```

3) Interface/HttpProxy/ProxyBackends/URLs/delete

Location within CLI Tree

CLI/Interface/HttpProxy/ProxyBackends/URLs

Command Format

```
delete          <backendName>, String {0 to 255 characters}
                <url>, String {0 to 255 characters}
```

Command Definition and Usage

```
                This command is used to remove a URL from an HTTP proxy backend.
```

Parameter Definitions

```
backendName     This parameter specifies the name of the backend.

url             This parameter specifies the URL prefix.
```

Example

```
                $ CLI/Interface/HttpProxy/ProxyBackends/URLs> delete backend1
                /path/to/webapp1
```

9.2.7 Interface/HttpProxy/ProxyBackends/Healthcheck

1) Interface/HttpProxy/ProxyBackends/Healthcheck/get

Location within CLI Tree

```
                CLI/Interface/HttpProxy/ProxyBackends/Healthcheck
```

Command Format

```
get             get takes no parameter.
```

Command Definition and Usage

```
                This command is used to view the health check settings of the HTTP
                proxy backend.
```

Example

```
                $ CLI/Interface/HttpProxy/ProxyServers/Healthcheck> get

                  Backend Name  In-band Fall  Out-of-band Rise  Out-of-band Fall
                Out-of-band Method  Out-of-band URL  Out-of-band Interval
                (seconds)  Out-of-band Timeout (seconds)
                ================================================================
                ================================================================
                ======================================
                          test             2                 3                 2
                              HEAD                 /
                        2                               2
```

2) Interface/HttpProxy/ProxyBackends/Healthcheck/set

Location within CLI Tree

CLI/Interface/HttpProxy/ProxyBackends/Healthcheck

Command Format

set	<attribute>, Multiple Choice = {inbandFall, outofbandRise, outofbandFall, outofbandMethod, outofbandUrl, outofbandInterval} <inbandFall>, Integer {0 to 100} <outofbandRise>, Integer {0 to 100} <outofbandFall>, Integer {0 to 100} <outofbandMethod>, Integer {1 to 255 characters} <outofbandUrl>, Integer {1 to 255 characters} <outofbandIntervalInSeconds>, Integer {1 to 300} <outofbandTimeoutInSeconds>, Integer {1 to 300}

Command Definition and Usage

This command is used to modify the health check settings of the HTTP proxy backend.

Parameter Definitions

inbandFall	This parameter specifies the number of consecutive failed in-band health checks that trigger a server to be considered offline.
outofbandRise	This parameter specifies the number of consecutive successful out-of-band health checks that trigger a server to be considered online.
outofbandFall	This parameter specifies the number of consecutive failed out-of-band health checks that trigger a server to be considered offline.
outofbandMethod	This parameter specifies the HTTP method that is used by the out-of-band health check requests.
outofbandUrl	This parameter specifies the URL that is used by the out-of-band health check requests.
outofbandIntervalInSeconds	This parameter specifies the number of seconds between two out-of-band health checks.
outofbandTimeoutInSeconds	This parameter specifies the maximum response time in seconds for the server to respond to a health check request.

Example

```
$ CLI/Interface/HttpProxy/ProxyServers/Healthcheck>
set inbandFall 3
```

9.2.8 Interface/HttpProxy/ProxyBindings

1) Interface/HttpProxy/ProxyBindings/get

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyBindings
```

Command Format

```
get             get takes no parameter
```

Command Definition and Usage

```
This command is used to view the HTTP proxy bindings.
```

Example

```
$ CLI/Interface/HttpProxy/ProxyBindings> get
  Interface Ip  Port   Backend
==============================
       1.2.3.4   443  backend1
```

2) Interface/HttpProxy/ProxyBindings/add

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyBindings
```

Command Format

```
add             <interfaceIp>, Choice = {<IP addresses>}
                <port>, Integer {0 to 65535}
                <backend>, String {0 to 255 characters}
```

Command Definition and Usage

```
This command is used to add a binding to the HTTP proxy.
```

Parameter Definitions

interfaceIp	This parameter specifies the interface IP address.
port	This parameter specifies the port number.
backend	This parameter specifies the name of the backend.

Example

```
$ CLI/Interface/HttpProxy/ProxyBindings> add 1.2.3.4 443 backend1
```

3) Interface/HttpProxy/ProxyBindings/delete

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyBindings
```

Command Format

```
delete          <interfaceIp>, Choice = {<IP addresses>}
                <port>, Integer {0 to 65535}
                <backend>, String {0 to 255 characters}
```

Command Definition and Usage

```
This command is used to remove the HTTP proxy's binding.
```

Parameter Definitions

interfaceIp	This parameter specifies the interface IP address.
port	This parameter specifies the port number.
backend	This parameter specifies the name of the backend.

Example

```
$ CLI/Interface/HttpProxy/ProxyBindings> delete 1.2.3.4 443
backend1
```

9.2.9 Interface/HttpProxy/ClientAuthentication/Trusts

1) Interface/HttpProxy/ClientAuthentication/Trusts/get

Location within CLI Tree

```
CLI/Interface/HttpProxy/ClientAuthentication/Trusts
```

Command Format

```
get             [<attribute>, Choice = {hideChain}]
```

Command Definition and Usage

```
This command is used to view the list of Trust Anchors.
```

Parameter Definitions

attribute	The name of the attribute to modify. Specify hideChain to prevent the chain of certificate authorities from being listed.

Example

```
$ CLI/Interface/HttpProxy/ClientAuthentication/Trusts> get
          Alias            Owner                          Issuer
================================================================
  http[auto-gen]  My Organization  My Organization[self-signed]
```

2) Interface/HttpProxy/ClientAuthentication/Trusts/createTrust

Location within CLI Tree

CLI/Interface/HttpProxy/ClientAuthentication/Trusts

Command Format

createTrust <alias>, String {1 to 64 characters}
[<attribute>, Multiple Choice = {keyLength, digest, country, stateOrProvince, city, organisationName, organisationUnit, emailAddress, challengePassword, commonName}]
<keyLength>, Integer {512 to 16384}
<digest>, Choice = {sha1, sha256}
<country>, String {2 characters}
<stateOrProvince>, String {1 to 50 characters}
<city>, String {1 to 50 characters}
<organisationName>, String {1 to 50 characters}
<organisationUnit>, String {1 to 50 characters}
<emailAddress>, String {1 to 50 characters}
<challengePassword>, String {1 to 40 characters}
<commonName>, String {1 to 255 characters}

Command Definition and Usage

This command is used to generate a Secure Sockets Layer (SSL) key and a trust anchor for a secure HTTP interface.

Parameter Definitions

alias	This parameter specifies administrator name to uniquely identify a trust anchor.
attribute	The name of the attribute to modify.
keyLength	This parameter specifies the key length or key size. (default value is "1024").
digest	This parameter specifies the message digest algorithm. (default value is "sha256").

country	This parameter specifies the country code international abbreviation code.
stateOrProvince	This parameter specifies the state or province.
city	This parameter specifies the name of the city.
organisationName	This parameter specifies the organization name.
organisationUnit	This parameter specifies the organization unit.
emailAddress	This parameter specifies the e-mail address.
challengePassword	This parameter specifies the challenge password.
commonName	This parameter specifies the common name. Generally, the common name on a client certificate is the proper name of the organization requesting a certificate. Alternatively, it may follow any standard already determined within the organization owning the certificate.

Example

```
$ CLI/Interface/HttpProxy/ClientAuthentication/Trusts>
createTrust alias digest sha1
```

3) Interface/HttpProxy/ClientAuthentication/Trusts/exportTrust

Location within CLI Tree

```
CLI/Interface/HttpProxy/ClientAuthentication/Trusts
```

Command Format

exportTrust

```
<alias>, String {1 to 64 characters}
<attribute>, Multiple Choice = {certificateFile, keyFile,
pkcs12File}
    pkcs12File:
        (pkcs12Password), Prompted password {5 to 255
characters}
```

Command Definition and Usage

This command is used to export a file related to Secure Sockets Layer (SSL). The SSL file is extracted from the BroadWorks trust store and copied to the */var/broadworks/tmp* directory.

Parameter Definitions

alias	This parameter specifies a name specified by the administrator to identify uniquely a trust anchor.

attribute	This name of the attribute to select the file type to export.
pkcs12File	This parameter exports the certificate and key in a Public-Key Cryptography Standards (PKCS#12) formatted keystore.
pkcs12Password	This parameter specifies the password used to secure the Public-Key Cryptography Standards (PKCS#12) formatted keystore.

Example

```
$ CLI/Interface/HttpProxy/ClientAuthentication/Trusts>
exportTrust alias certificateFile
```

4) Interface/HttpProxy/ClientAuthentication/Trusts/showTrust

Location within CLI Tree

```
CLI/Interface/HttpProxy/ClientAuthentication/Trusts
```

Command Format

showTrust	<alias>, String {1 to 64 characters}

Command Definition and Usage

```
This command is used to view the details of the X.509
certificate associated with the trust anchor of the provided
interface.
```

Parameter Definitions

alias	This parameter specifies a name to uniquely identify a trust anchor.

Example

```
$ CLI/Interface/HttpProxy/ClientAuthentication/Trusts>
showTrust Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 16741255617481617075
(0xe854e348fe012eb3)
    Signature Algorithm: sha256WithRSAEncryption
...
```

5) Interface/HttpProxy/ClientAuthentication/Trusts/updateTrust

Location within CLI Tree

CLI/Interface/HttpProxy/ClientAuthentication/Trusts

Command Format

updateTrust	<alias>, String {1 to 64 characters} <trustAnchorFile>, String {1 to 255 characters} [<trustAnchorChain>, String {1 to 255 characters}]

Command Definition and Usage

This command is used to load and update a trust anchor file. The certificate is taken from the specified path and copied to the proper location. If a trust anchor chain is provided, it is used for determining the certificate hierarchy when storing the trust anchor.

Parameter Definitions

alias	This parameter specifies a name to uniquely identify a trust anchor.
trustAnchorFile	This parameter specifies the filename of the trust anchor file.
trustAnchorChain	This parameter specifies the filename of the trust anchor chain.

Example

```
$ CLI/Interface/HttpProxy/ClientAuthentication/Trusts>
updateTrust alias /path/to/trustAnchorFile
```

9.2.10 Interface/HttpProxy/ProxyAlias

1) Interface/HttpProxy/ProxyAlias/get

Location within CLI Tree

CLI/Interface/HttpProxy/ProxyAlias

Command Format

get	get takes no parameter

Command Definition and Usage

This command is used to retrieve the list of Hypertext Transfer Protocol (HTTP) proxy aliases.

Example

```
$ XSP_CLI/Interface/HttpProxy/ProxyAliases> g
  Interface IP        Name
==========================
       1.2.3.4        test
```

2) Interface/HttpProxy/ProxyAlias/add

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyAlias
```

Command Format

```
add             <interfaceIp>, Choice = {<IP addresses>}
                <name>, IP address | host | domain (1 to 80 chars)
```

Command Definition and Usage

```
This command is used to add a Hypertext Transfer Protocol
(HTTP) proxy alias for a physical interface.
```

Parameter Definitions

interfaceIp	This parameter specifies the interface Internet Protocol (IP) address.
name	This parameter specifies the name of the Hypertext Transfer Protocol (HTTP) alias.

Example

```
$ CLI/Interface/HttpProxy/ProxyAlias> add 1.2.3.4 test
```

3) Interface/HttpProxy/ProxyAlias/delete

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyAlias
```

Command Format

```
delete          <interfaceIp>, Choice = {<IP addresses>}
                <name>, IP address | host | domain (1 to 80 chars)
```

Command Definition and Usage

```
This command is used to delete a Hypertext Transfer Protocol
(HTTP) proxy alias for an interface.
```

Parameter Definitions

interfaceIp	This parameter specifies the interface Internet Protocol (IP) address.
name	This parameter specifies the name of the Hypertext Transfer Protocol (HTTP) server.

Example

```
$ CLI/Interface/HttpProxy/ProxyAlias> delete 1.2.3.4 test
```

4) Interface/HttpProxy/ProxyAlias/sslExport

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyAlias
```

Command Format

sslExport	<alias>, IP address | host | domain (1 to 80 chars) <attribute>, Multiple Choice = {certificateFile, keyFile, chainFile}

Command Definition and Usage

This command is used to export Secure Sockets Layer (SSL) files (certificate, key, or chain file). The SSL file is taken from its location and copied to the */var/broadworks/tmp/* directory.

Parameter Definitions

alias	This parameter specifies the name of the alias.
attribute	The name of the file to export.

Example

```
$ CLI/Interface/HttpProxy/ProxyAlias> sslExport 1.2.3.4
certificateFile
```

5) Interface/HttpProxy/ProxyAlias/sslGenkey

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyAlias
```

Command Format

sslGenkey	<alias>, IP address | host | domain (1 to 80 chars) [<attribute>, Multiple Choice = {keyLength, digest, country, stateOrProvince, city, organisationName, organisationUnit, emailAddress, challengePassword, commonName}] <keyLength>, Integer {512 to 16384} <digest>, Choice = {sha1, sha256} <country>, String {2 characters} <stateOrProvince>, String {1 to 50 characters} <city>, String {1 to 50 characters} <organisationName>, String {1 to 50 characters} <organisationUnit>, String {1 to 50 characters} <emailAddress>, String {1 to 50 characters} <challengePassword>, String {1 to 40 characters} <commonName>, String {1 to 255 characters}

Command Definition and Usage

This command is used to generate a Secure Sockets Layer (SSL) key for an alias. A generated Certificate Signing Request (*.csr*) file is written to the */tmp* directory under the *name.csr*, where name is the interface IP address.

Parameter Definitions

alias	This parameter specifies the name of the alias.
attribute	The name of the attribute to set.
keyLength	This parameter specifies the SSL key length or key size. The default value (when not specified) is "1024".
digest	This parameter specifies the message digest algorithm. The default value (when not specified) is "sha256".
country	This parameter specifies the country code international abbreviation code.
stateOrProvince	This parameter specifies the state or province.
city	This parameter specifies the name of the city.
organisationName	This parameter specifies the organization name.
organisationUnit	This parameter specifies the organization unit.
emailAddress	This parameter specifies the e-mail address.
challengePassword	This parameter specifies the challenge password.

commonName	This parameter specifies the common name. It must correspond to the name of this proxy alias. A "*" wildcard character may be present in the left-most name component of the host if the certificate is used on multiple servers having the same domain. For example, the common name for host xsp1.broadsoft.com could be *.broadsoft.com or xsp*.broadsoft.com.

Example

```
$ CLI/Interface/HttpProxy/ProxyAlias> sslGenkey 1.2.3.4
digest sha256
```

6) Interface/HttpProxy/ProxyAlias/sslRemove

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyAlias
```

Command Format

sslRemove	<alias>, IP address | host | domain (1 to 80 chars) <attribute>, Multiple Choice = {chainFile}

Command Definition and Usage

```
This command is used to remove the Secure Sockets Layer
(SSL) file (chain file).
```

Parameter Definitions

alias	This parameter specifies the name of the alias.
attribute	The type of file to remove.

Example

```
$ CLI/Interface/HttpProxy/ProxyAlias> sslRemove 1.2.3.4
chainFile
```

7) Interface/HttpProxy/ProxyAlias/sslShow

Location within CLI Tree

```
CLI/Interface/HttpProxy/ProxyAlias
```

Command Format

sslShow	<alias>, IP address | host | domain (1 to 80 chars)

Command Definition and Usage

This command is used to view the details of the X.509 certificate and certificate chain associated with the provided alias. This command reflects the information sent by the server to the client software.

Parameter Definitions

alias	This is the name of the alias.

Example

```
$ CLI/Interface/HttpProxy/ProxyAlias> sslShow 1.2.3.4
```

8) Interface/HttpProxy/ProxyAlias/sslUpdate

Location within CLI Tree

CLI/Interface/HttpProxy/ProxyAlias

Command Format

sslUpdate <alias>, IP address | host | domain (1 to 80 chars)
<attribute>, Multiple Choice = {certificateFile, keyFile, chainFile}
<certificateFile>, String {1 to 255 characters}
<keyFile>, String {1 to 255 characters}
<chainFile>, String {1 to 255 characters}

Command Definition and Usage

This command is used to load and update a Secure Sockets Layer (SSL) certificate file. The certificate is taken from the specified path and copied to the proper location.

Parameter Definitions

alias	This parameter specifies the name of the alias.
attribute	The name of a Secure Sockets Layer (SSL) configuration attribute to modify.
certificateFile	This parameter specifies the file name of the certificate file.
keyFile	This parameter specifies the file name of the key file.
chainFile	This parameter specifies the file name of the certificate chain file.

Example

```
$ CLI/Interface/HttpProxy/ProxyAliass> sslUpdate 1.2.3.4
certificateFile /path/to/certificate.cert
```

9.2.11 Applications/LoadBalancer/HAProxy/GeneralSettings

1) Applications/LoadBalancer/HAProxy/GeneralSettings/get

Location within CLI Tree

```
CLI/Applications/LoadBalancer/HAProxy/GeneralSettings
```

Command Format

```
get             get takes no parameter.
```

Command Definition and Usage

```
This command is used to view the load balancer general settings.
```

Example

```
$ CLI/Applications/LoadBalancer/HAProxy/GeneralSettings> get
  nbProc = 1
  statsPortRangeStart = 9000
```

2) Applications/LoadBalancer/HAProxy/GeneralSettings/set

Location within CLI Tree

```
CLI/Applications/LoadBalancer/HAProxy/GeneralSettings
```

Command Format

```
set                     <attribute>, Multiple Choice = {nbProc,
                        statPortRangeStart, logPort }
                            <nbProc>, Integer {1 to 64}
                            <statPortRangeStart>, Integer {1025 to 65535}
```

Command Definition and Usage

```
This command is used to modify the load balancer
general settings.
```

Parameter Definitions

```
nbProc                  This parameter specifies the number of load balancer
                        processes that are spawned at application startup.
```

statPortRangeStart	This parameter specifies the start of the TCP port range used by the load balancer to expose its statistics. The length of the range is equal to the number of processes.

Example

```
$
CLI/Applications/LoadBalancer/HAProxy/GeneralSettings>
set nbProc 1
```

9.2.12 Applications/LoadBalancer/HAProxy/AutomaticReload

1) Applications/LoadBalancer/HAProxy/AutomaticReload/get

Location within CLI Tree

```
CLI/Applications/LoadBalancer/HAProxy/AutomaticReload
```

Command Format

get	get takes no parameter.

Command Definition and Usage

This command is used to view the HAProxy automatic configuration reload settings.

Example

```
$ CLI/Applications/LoadBalancer/HAProxy/AutomaticReload> get
  enabled = true
  delayInSeconds = 30
```

2) Applications/LoadBalancer/HAProxy/AutomaticReload/set

Location within CLI Tree

```
CLI/Applications/LoadBalancer/HAProxy/AutomaticReload
```

Command Format

set	<attribute>, Multiple Choice = {enabled, delayInSeconds} <enabled>, Choice = {false, true} <delayInSeconds>, Integer {1 to 3600}

Command Definition and Usage

This command is used to modify the HAProxy automatic configuration reload settings.

Parameter Definitions

enabled	This parameter controls the automatic configuration reload.
delayInSecond	This parameter specifies the delay after the last configuration change when the load balancer configuration is reloaded automatically.

Example

```
$
CLI/Applications/LoadBalancer/HAProxy/AutomaticReload>
set enabled false
```

9.2.13 Applications/LoadBalancer/HAProxyController/GeneralSettings

1) Applications/LoadBalancer/HAProxyController/GeneralSettings/get

Location within CLI Tree

```
CLI/Applications/LoadBalancer/HAProxyController/GeneralSettings
```

Command Format

get	get takes no parameter.

Command Definition and Usage

This command is used to view the load balancer controller general settings.

Example

```
$ CLI/Applications/LoadBalancer/HAProxyController/GeneralSettings>
get
  logPort = 8000
```

2) Applications/LoadBalancer/HAProxyController/GeneralSettings/set

Location within CLI Tree

```
CLI/Applications/LoadBalancer/HAProxyController/GeneralSettings
```

Command Format

set	<attribute>, Multiple Choice = {nbProc, statPortRangeStart, logPort } <logPort>, Integer {1025 to 65535}

Command Definition and Usage

This command is used to modify the load balancer controller general settings.

Parameter Definitions

logPort	This parameter specifies the UDP port used by the load balancer controller to receive the load balancer logs.

Example

```
$
CLI/Applications/LoadBalancer/HAProxyController/GeneralSettings>
set logPort 9001
```

9.2.14 Applications/WebContainer/Tomcat/GeneralSettings

1) Applications/WebContainer/Tomcat/GeneralSettings/get

Location within CLI Tree

CLI/Applications/WebContainer/Tomcat/GeneralSettings

Command Format

get	get takes no parameter.

Command Definition and Usage

This command is used to view the Tomcat general settings.

Example

```
$ CLI/Applications/WebContainer/Tomcat/GeneralSettings> get
  uriEncoding = UTF-8
  authenticationEncoding = ISO-8859-1
  statisticsRefreshPeriod = 5
  maxHttpHeaderSize = 1024
  xFrameOptionsSameOrigin = true
  jvmRoute =
```

2) Applications/WebContainer/Tomcat/GeneralSettings/set

Location within CLI Tree

CLI/Applications/WebContainer/Tomcat/GeneralSettings

Command Format

set <attribute>, Multiple Choice = {uriEncoding, authenticationEncoding, statisticsRefreshPeriod, maxHttpHeaderSize, xFrameOptionsSameOrigin, jvmRoute, certificateInHeaders}
<uriEncoding>, Choice = {UTF-8, ISO-8859-1}
<authenticationEncoding>, Choice = {UTF-8, ISO-8859-1}
<statisticsRefreshPeriod>, Integer {1 to 86400}
<maxHttpHeaderSize>, Integer {1 to 32768}
<xFrameOptionsSameOrigin>, Choice = {false, true}
<jvmRoute>, String {1 to 80 characters}

Command Definition and Usage

This command is used to modify Tomcat general settings.

Parameter Definitions

attribute	The name of an attribute to modify.
uriEncoding	This parameter specifies the character encoding for URI definition.
authenticationEncoding	This parameter specifies the character encoding used decode the username and password coming from the header when using basic authentication.
statisticsRefreshPeriod	This parameter specifies the frequency at which BroadWorks refreshes PMs within Tomcat.
maxHttpHeaderSize	This parameter specifies the limit (in bytes) on the allowed size of an HTTP request header field.
xFrameOptionsSameOrigin	This parameter enables the addition of the X-Frame-Options HTTP Header (defined by RFC 7034) to all HTTP responses, with a value of "SAMEORIGIN". This is used to improve the protection of web applications against clickjacking.
jvmRoute	This parameter specifies the name of this server instance that will be appended to the JSESSIONID in HTTP responses.

Example

```
$
CLI/Applications/WebContainer/Tomcat/GeneralSettings>
set jvmRoute serverA
```

3) Applications/WebContainer/Tomcat/GeneralSettings/clear

Location within CLI Tree

CLI/Applications/WebContainer/Tomcat/GeneralSettings

Command Format

clear	<attribute>, Multiple Choice = {maxHttpHeaderSize, jvmRoute}

Command Definition and Usage

This command is used to modify Tomcat general settings.

Parameter Definitions

attribute	The name of the attribute(s) to clear.

Example

```
$
CLI/Applications/WebContainer/Tomcat/GeneralSettings>
clear jvmRoute
```

9.2.15 Applications/WebContainer/Tomcat/ClientAuthenticationProxy

1) Applications/WebContainer/Tomcat/ClientAuthenticationProxy/get

Location within CLI Tree

CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy

Command Format

get	get takes no parameter.

Command Definition and Usage

This command is used to view the client authentication proxy settings.

Example

```
$ CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy>
get
  enable = true
```

2) Applications/WebContainer/Tomcat/ClientAuthenticationProxy/set

Location within CLI Tree

```
CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy
```

Command Format

```
set             <attribute>, Multiple Choice = {enable}
                    <enable>, Choice = {false, true}
```

Command Definition and Usage

```
This command is used to modify the client authentication proxy
settings.
```

Parameter Definitions

attribute	The name of an attribute to modify.
enable	This parameter controls whether the client certificate information located in the HTTP headers will be parsed and forwarded to the web applications. Enabling this option is useful when the client SSL connection is terminated by an external component.

Example

```
$
CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy>
set enable true
```

9.2.16 Applications/WebContainer/Tomcat/ClientAuthenticationProxy/NetworkAccessList

1) Applications/WebContainer/Tomcat/ClientAuthenticationProxy/NetworkAccessList/get

Location within CLI Tree

```
CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy/Networ
kAccessList
```

Command Format

```
get         get takes no parameter.
```

Command Definition and Usage

```
This command is used to view the network access list settings.
```

Example

```
$
CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy/Networ
kAccessList> get
     Address  Description
=========================
     1.2.3.4
```

2) Applications/WebContainer/Tomcat/ClientAuthenticationProxy/NetworkAccessList/add

Location within CLI Tree

```
CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy/Ne
tworkAccessList
```

Command Format

```
add             <address>, IP address (1 to 39 chars)
                [<attribute>, Multiple Choice = {description}]
                    <description>, String {1 to 80 characters}
```

Command Definition and Usage

```
This command is used to add a network access list entry.
```

Parameter Definitions

address	This parameter specifies the Internet Protocol (IP) address of the network element that sends client authentication information to the web container.
attribute	The name of an attribute to add.
description	This parameter specifies an optional description for the network element.

Example

```
$
CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy/Ne
tworkAccessList> set enable true
```

3) Applications/WebContainer/Tomcat/ClientAuthenticationProxy/NetworkAccessList/set

Location within CLI Tree

```
CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy/Ne
tworkAccessList
```

Command Format

```
set             <address>, IP address (1 to 39 chars)
                [<attribute>, Multiple Choice = {description}]
                    <description>, String {1 to 80 characters}
```

Command Definition and Usage

This command is used to modify a network access list entry.

Parameter Definitions

address	This parameter specifies the Internet Protocol (IP) address of the network element that sends client authentication information to the web container.
attribute	The name of an attribute to set.
description	This parameter specifies an optional description for the network element.

Example

```
$
CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy/Ne
tworkAccessList> set 1.2.3.4 description test
```

4) Applications/WebContainer/Tomcat/ClientAuthenticationProxy/NetworkAccessList/delete

Location within CLI Tree

```
CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy/Ne
tworkAccessList
```

Command Format

```
delete          <address>, IP address (1 to 39 chars)
```

Command Definition and Usage

This command is used to delete a network access list entry.

Parameter Definitions

address	This parameter specifies the Internet Protocol (IP) address of the network element that sends client authentication information to the web container.

Example

```
$
CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy/Ne
tworkAccessList> delete 1.2.3.4
```

5) Applications/WebContainer/Tomcat/ClientAuthenticationProxy/NetworkAccessList/clear

Location within CLI Tree

```
CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy/Ne
tworkAccessList
```

Command Format

```
clear          <address>, IP address (1 to 39 chars)
               <attribute>, Multiple Choice = {description}
```

Command Definition and Usage

```
This command is used to clear (set to no value) the network
access list attributes.
```

Parameter Definitions

address	This parameter specifies the Internet Protocol (IP) address of the network element that sends client authentication information to the web container.
attribute	The name of an attribute to clear.

Example

```
$
CLI/Applications/WebContainer/Tomcat/ClientAuthenticationProxy/Ne
tworkAccessList> clear 1.2.3.4 description
```

9.2.17 Applications/LogServer/GeneralSettings

1) Applications/LogServer/GeneralSettings/get

Location within CLI Tree

```
CLI/Applications/LogServer/GeneralSettings
```

Command Format

```
get            get takes no parameter.
```

Command Definition and Usage

```
This command is used to modify the general settings of the log
server.
```

Example

```
$ CLI/Applications/LogServer/generalSettings> get
  udpPort = 4444
  syslogUdpPort = 4445
```

2) Applications/LogServer/GeneralSettings/set

Location within CLI Tree

```
CLI/Applications/LogServer/GeneralSettings
```

Command Format

```
set                 <attribute>, Multiple Choice = {udpPort, syslogUdpPort}
                        <udpPort>, Integer {0 to 65535}
                        <syslogUdpPort>, Integer {0 to 65535}
```

Command Definition and Usage

```
This command is used to modify the general settings of the log
server.
```

Parameter Definitions

attribute	The name of an attribute to modify.
udpPort	This parameter specifies the log server port.
syslogUdpPort	This parameter specifies the syslog server port.

Example

```
$ CLI/Applications/LogServer/generalSettings> set
syslogUdpPort 4445
```

9.3 Open Client Interface-Provisioning Impact

There is no impact.

9.4 External Authentication Impacts

There is no impact.

9.5 Application Server Portal API Impacts

There is no impact.

9.6 Network Server Location API Impacts

There is no impact.

9.7 NSSync API Impacts

There is no impact.

9.8 Application Server Dump Impacts

There is no impact.

9.9 BroadCloud Dump Impacts

There is no impact.

9.10 Service License Reporting Impact

There is no impact.

9.11 Call Detail Server SOAP Interface

There is no impact.

9.12 Treatments

This is not applicable.

9.13 Media Announcements (Audio and Video)

This is not applicable.

9.14 BroadWorks Common Communication Transport Impacts

There is no impact.

9.15 Device Management Impacts

There is no impact.

10 Accounting Impacts

There is no impact.

11 System Management Impacts

11.1 Performance Management Impacts

A new management information base (MIB) is added for the Load Balancer: *BW-LoadBalancer.mib*.

11.1.1 New Counters

The new counters in the following tables are added.

11.1.1.1 ProxyServers Module

The *proxyServers* module table contains the following columns.

Name:	bwLoadBalancerProxyServerName
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This string contains the name of the Load Balancer proxy server.
Type:	DisplayString
Access:	read-only
Incremented:	This string is set when the Load Balancer is started.

Name:	bwLoadBalancerProxyServerCurrentConnections
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This gauge tracks the number of active sessions.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is incremented when a new client connection is established and it is decremented when a client connection is closed.

Name:	bwLoadBalancerProxyServerMaxConnections
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This counter tracks the maximum number of active sessions since the last restart of the Load Balancer application.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is incremented when a new high-water mark of the number of active connections is reached.

Name:	bwLoadBalancerProxyServerLimitConnections
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This gauge tracks the configured limit of the number of active connections.
Type:	Gauge32
Access:	read-only

Incremented:	This gauge is set when the Load Balancer Is started.

Name:	bwLoadBalancerProxyServerTotalConnections
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This counter tracks the cumulative number of active connections.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a new client connection is established.

Name:	bwLoadBalancerProxyServerBytesIn
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This counter tracks the number of bytes received from the client to this proxy server.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP request is received from a client.

Name:	bwLoadBalancerProxyServerBytesOut
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This counter tracks the number of bytes sent to the client from this proxy server.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response is sent to a client.

Name:	bwLoadBalancerProxyServerRequestsDenied
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This counter tracks the number of requests denied by this proxy server.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP request is rejected by this frontend.

Name:	bwLoadBalancerProxyServerRequestErrors
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This counter tracks the number of request errors.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP request was not received correctly.

Name:	bwLoadBalancerProxyServerConnectionRate

MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This gauge tracks the number of new connections over the last elapsed second.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated continually.

Name:	bwLoadBalancerProxyServerMaxConnectionRate
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This gauge tracks the maximum number of new sessions per second since the last restart of the Load Balancer application.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated when the high-water mark of the connection rate increases.

Name:	bwLoadBalancerProxyServer1xxResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This counter tracks the number of HTTP responses containing a status code from 100 through 199.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 100 through 199 is sent to a client.

Name:	bwLoadBalancerProxyServer2xxResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This counter tracks the number of HTTP responses containing a status code from 200 through 299.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 200 through 299 is sent to a client.

Name:	bwLoadBalancerProxyServer3xxResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This counter tracks the number of HTTP responses containing a status code from 300 through 399.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 300 through 399 is sent to a client.

Name:	bwLoadBalancerProxyServer4xxResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This counter tracks the number of HTTP responses containing a status code from 400 through 499.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 400 through 499 is sent to a client.

Name:	bwLoadBalancerProxyServer5xxResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This counter tracks the number of HTTP responses containing a status code from 500 through 599.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 500 through 599 is sent to a client.

Name:	bwLoadBalancerProxyServerOtherResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This counter tracks the number of HTTP responses containing a status code outside the 100 through 599 range.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code starting with a number outside the 100 through 599 range is sent to a client.

Name:	bwLoadBalancerProxyServerRequestRate
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This gauge tracks the number of requests received over the last elapsed second.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated continually.

Name:	bwLoadBalancerProxyServerMaxRequestRate
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This gauge tracks the maximum number of requests received per second since the last restart of the Load Balancer application.
Type:	Gauge32
Access:	read-only

Incremented:	This gauge is incremented when a new high-water mark of the number of requests per second is reached.

Name:	bwLoadBalancerProxyServerRequestsTotal
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.proxyServers
Description:	This counter tracks the number of requests received.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a new HTTP request is received from a client.

11.1.1.2 Backends Module

The *backends* module consists of two tables. The first table contains general backend performance information. This information is gathered from all Load Balancer processes and is coalesced into a common view. The second table contains backend status information as updated by the health check mechanism. This information is presented separately for each Load Balancer process.

The *backends* module table contains the following columns.

Name:	bwLoadBalancerBackendName
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This string contains the name of the Load Balancer backend.
Type:	DisplayString
Access:	read-only
Incremented:	This string is set when the Load Balancer is started.

Name:	bwLoadBalancerBackendCurrentQueue
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This gauge tracks the number of queued requests without a server assigned.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is incremented when a request is queued in the backend and it is decremented when a request is assigned to a server.

Name:	bwLoadBalancerBackendMaxQueue
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This gauge tracks the maximum number of queued requests without a server assigned since the last restart of the Load Balancer application.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is incremented when a new high-water mark of the number of queued requests is reached.

Name:	bwLoadBalancerBackendCurrentConnections
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This gauge tracks the number of active connections.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is incremented when a new client connection is established and it is decremented when a client connection is closed.

Name:	bwLoadBalancerBackendMaxConnections
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This gauge tracks the maximum number of active connections reached since the last restart of the Load Balancer application.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is incremented when a new high-water mark of the number of active connections is reached.

Name:	bwLoadBalancerBackendLimitConnections
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This gauge tracks the configured limit of the number of active connections.
Type:	Gauge32
Access:	read only
Incremented:	This gauge is set when the Load Balancer is started.

Name:	bwLoadBalancerBackendTotalConnections
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the cumulative number of active connections.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a new client connection is established.

Name:	bwLoadBalancerBackendBytesIn
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of bytes received from a proxy server to this backend.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP request is received from a frontend.

Name:	bwLoadBalancerBackendBytesOut
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of bytes sent to a proxy server from this backend.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response is sent to a frontend.

Name:	bwLoadBalancerBackendRequestsDenied
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of requests denied by this backend.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP request is rejected by this frontend.

Name:	bwLoadBalancerBackendResponsesDenied
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of responses denied to be proxied back to the client.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response is denied.

Name:	bwLoadBalancerBackendConnectionErrors
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of connection errors of this backend to its servers.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a connection fails to be established to a backend server.

Name:	bwLoadBalancerBackendResponseErrors
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of response errors from servers.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a response fails to be received from a server.

Name:	bwLoadBalancerBackendRetries
MIB:	BW-LoadBalancer.mib

Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of times a connection to a server is retried.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when the Load Balancer fails to connect to a server and tries again.

Name:	bwLoadBalancerBackendRedispatches
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of times the Load Balancer redispatches a request to a different server after too many failed connection attempts.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a request is redispatched to a different server.

Name:	bwLoadBalancerBackendLoadBalanced
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of times the Load Balancer algorithm is used to assign a server to a session.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when the load balancing algorithm is used.

Name:	bwLoadBalancerBackendConnectionRate
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This gauge tracks the number of new connections over the last elapsed second assigned to this backend.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated continually.

Name:	bwLoadBalancerBackendMaxConnectionRate
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This gauge tracks the maximum number of new connections per second since the last restart of the Load Balancer application.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated when the high-water mark of the connection rate increases.

Name:	bwLoadBalancerBackend1xxResponses
MIB:	BW-LoadBalancer.mib

Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of HTTP responses containing a status code from 100 through 199.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 100 through 199 is sent to a client.

Name:	bwLoadBalancerBackend2xxResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of HTTP responses containing a status code from 200 through 299.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 200 through 299 is sent to a client.

Name:	bwLoadBalancerBackend3xxResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of HTTP responses containing a status code from 300 through 399.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 300 through 399 is sent to a client.

Name:	bwLoadBalancerBackend4xxResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of HTTP responses containing a status code from 400 through 499.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 400 through 499 is sent to a client.

Name:	bwLoadBalancerBackend5xxResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of HTTP responses containing a status code from 500 through 599.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 500 through 599 is sent to a client.

Name:	bwLoadBalancerBackendOtherResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of HTTP responses containing a status code outside the 100 through 599 range.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code starting with a number outside the 100 through 599 range is sent to a client.

Name:	bwLoadBalancerBackendClientAborts
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of data transfers aborted by the client.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a data transfer is aborted by the client.

Name:	bwLoadBalancerBackendServerAborts
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter tracks the number of data transfers aborted by the server.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a data transfer is aborted by the server.

Name:	bwLoadBalancerBackendLastConnectionTime
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This date tracks the time when the last connection was assigned to this backend.
Type:	DateAndTime
Access:	read-only
Incremented:	This date is updated when a connection is assigned to this backend.

Name:	bwLoadBalancerBackendQueueTime
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This gauge tracks the average queue time in milliseconds of the last 1024 requests that passed through this backend.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated when a response is received from a server.

Name:	bwLoadBalancerBackendConnectTime
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This gauge tracks the average connect time in milliseconds of the last 1024 requests that passed through this backend.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated when a response is received from a server.

Name:	bwLoadBalancerBackendResponseTime
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This gauge tracks the average response time in milliseconds of the last 1024 requests that passed through this backend.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated when a response is received from a server.

Name:	bwLoadBalancerBackendTotalTime
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This gauge tracks the average total session time in milliseconds of the last 1024 requests that passed through this backend.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated when a response is received from a server.

The *backends* module status table contains the following columns.

Name:	bwLoadBalancerBackendStatusProcessId
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This string identifies the process hosting the Load Balancer backend.
Type:	DisplayString
Access:	read-only
Incremented:	This string is set when the Load Balancer is started.

Name:	bwLoadBalancerBackendStatusName
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This string contains the name of the Load Balancer backend.
Type:	DisplayString
Access:	read-only
Incremented:	This string is set when the Load Balancer is started.

Name:	bwLoadBalancerBackendStatusStatus
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This string contains the current status of the Load Balancer backend.
Type:	DisplayString
Access:	read-only
Incremented:	This string is modified when the status of the backend changes.

Name:	bwLoadBalancerBackendStatusDownChecks
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter contains the number of times the backend was transitioned to the down state following failed health checks.
Type:	Counter64
Access:	read-only
Incremented:	This counter is incremented when the backend transitions from the up state to the down state.

Name:	bwLoadBalancerBackendStatusLastChange
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This date contains the last time a change in the status of this backend occurred.
Type:	DateAndTime
Access:	read-only
Incremented:	This date is updated when the backend state changes.

Name:	bwLoadBalancerBackendStatusDowntime
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backends
Description:	This counter contains the cumulative number of seconds this backend was in the down state.
Type:	Counter64
Access:	read-only
Incremented:	This counter is incremented while the backend is in the down state.

11.1.1.3 Backend Servers Module

The *backendServers* module consists of two tables. The first table contains general backend server performance information. This information is gathered from all Load Balancer processes and is coalesced into a common view. The second table contains backend status information as updated by the health check mechanism. This information is presented separately for each Load Balancer process. The *backendServers* module table contains the following columns.

Name:	bwLoadBalancerBackendServerName
MIB:	BW-LoadBalancer.mib

Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This string contains the name of the Load Balancer server.
Type:	DisplayString
Access:	read-only
Incremented:	This string is set when the Load Balancer is started.

Name:	bwLoadBalancerBackendServerCurrentQueue
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This gauge tracks the number of queued requests.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is incremented when a request is queued in the backend and it is decremented when a request is assigned to a server.

Name:	bwLoadBalancerBackendServerMaxQueue
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This gauge tracks the maximum number of queued requests since the last restart of the Load Balancer application.
Type:	Gauge32
Access:	read-only
Incremented:	This counter is incremented when a new high-water mark of the number of queued requests is reached.

Name:	bwLoadBalancerBackendServerCurrentConnections
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This gauge tracks the number of active connections.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is incremented when a new client connection is established and it is decremented when a client connection is closed.

Name:	bwLoadBalancerBackendServerMaxConnections
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This gauge tracks the maximum number of active connections since the last restart of the Load Balancer application.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is incremented when a new high-water mark of the number of active sessions is reached.

Name:	bwLoadBalancerBackendServerLimitConnections

MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This gauge tracks the configured limit of the number of active connections.
Type:	Gauge32
Access:	read only
Incremented:	This gauge is set when the Load Balancer is started.

Name:	bwLoadBalancerBackendServerTotalConnections
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the cumulative number of active connections.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a new client connection is established.

Name:	bwLoadBalancerBackendServerBytesIn
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of bytes received from a backend to this server.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP request is received from a backend.

Name:	bwLoadBalancerBackendServerBytesOut
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of bytes sent to a backend from this server.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response is sent to a backend.

Name:	bwLoadBalancerBackendServerResponsesDenied
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of responses denied to be proxied back to the client.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response is denied.

Name:	bwLoadBalancerBackendServerConnectionErrors
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers

Description:	This counter tracks the number of connection errors to this server.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a connection fails to be established to a backend server.

Name:	bwLoadBalancerBackendServerResponseErrors
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of response errors from this server.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a response fails to be received from a server.

Name:	bwLoadBalancerBackendServerRetries
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of a connection to this server is retried.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when the Load Balancer fails to connect to a server and tries again.

Name:	bwLoadBalancerBackendServerRedispatches
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of times the Load Balancer redispatches a request away from this server after too many failed connection attempts.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a request is redispatched to a different server.

Name:	bwLoadBalancerBackendServerCheckFails
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of times a health check to this server failed.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a health check fails.

Name:	bwLoadBalancerBackendServerLoadBalanced
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers

Description:	This counter tracks the number of times the Load Balancer algorithm is used to assign this server to a connection.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when the load balancing algorithm is used.

Name:	bwLoadBalancerBackendServerConnectionRate
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This gauge tracks the number of new connections over the last elapsed second assigned to this server.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated continually.

Name:	bwLoadBalancerBackendServerMaxConnectionRate
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This gauge tracks the maximum number of new connections per second since the last restart of the Load Balancer application.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated when the high-water mark of the connection rate increases.

Name:	bwLoadBalancerBackendServer1xxResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of HTTP responses containing a status code from 100 through 199.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 100 through 199 is sent to a client.

Name:	bwLoadBalancerBackendServer2xxResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of HTTP responses containing a status code from 200 through 299.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 200 through 299 is sent to a client.

Name:	bwLoadBalancerBackendServer3xxResponses
MIB:	BW-LoadBalancer.mib

Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of HTTP responses containing a status code from 300 through 399.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 300 through 399 is sent to a client.

Name:	bwLoadBalancerBackendServer4xxResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of HTTP responses containing a status code from 400 through 499.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 400 through 499 is sent to a client.

Name:	bwLoadBalancerBackendServer5xxResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of HTTP responses containing a status code from 500 through 599.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code from 500 through 599 is sent to a client.

Name:	bwLoadBalancerBackendServerOtherResponses
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of HTTP responses containing a status code outside the 100 through 599 range.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when an HTTP response with a status code starting with a number outside the 100 through 599 range is sent to a client.

Name:	bwLoadBalancerBackendServerClientAborts
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of data transfers aborted by the client.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a data transfer is aborted by the client.

Name:	bwLoadBalancerBackendServerServerAborts
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter tracks the number of data transfers aborted by the server.
Type:	Counter32
Access:	read-only
Incremented:	This counter is incremented when a data transfer is aborted by the server.

Name:	bwLoadBalancerBackendServerLastConnectionTime
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This date tracks the time when the last connection was assigned to this server.
Type:	DateAndTime
Access:	read-only
Incremented:	This date is updated when a connection is assigned to this backend.

Name:	bwLoadBalancerBackendServerQueueTime
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This gauge tracks the average queue time in milliseconds of the last 1024 requests that passed through this server.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated when a response is received from a server.

Name:	bwLoadBalancerBackendServerConnectTime
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This gauge tracks the average connect time in milliseconds of the last 1024 requests that passed through this server.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated when a response is received from a server.

Name:	bwLoadBalancerBackendServerResponseTime
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This gauge tracks the average response time in milliseconds of the last 1024 requests that passed through this server.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated when a response is received from a server.

Name:	bwLoadBalancerBackendServerTotalTime

MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This gauge tracks the average total session time in milliseconds of the last 1024 requests that passed through this server.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated when a response is received from a server.

The *backendServers* module status table contains the following columns.

Name:	bwLoadBalancerBackendServerStatusProcessId
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This string identifies the process hosting the Load Balancer backend server.
Type:	DisplayString
Access:	read-only
Incremented:	This string is set when the Load Balancer is started.

Name:	bwLoadBalancerBackendServerStatusName
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This string contains the name of the Load Balancer backend server.
Type:	DisplayString
Access:	read-only
Incremented:	This string is set when the Load Balancer is started.

Name:	bwLoadBalancerBackendServerStatusStatus
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This string contains the current status of the Load Balancer backend server.
Type:	DisplayString
Access:	read-only
Incremented:	This string is modified when the status of the backend server changes.

Name:	bwLoadBalancerBackendServerStatusFailedChecks
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter contains the number of failed health checks.
Type:	Counter64
Access:	read-only
Incremented:	This counter is incremented when a health check is not successful.

Name:	bwLoadBalancerBackendServerStatusDownChecks
MIB:	BW-LoadBalancer.mib

Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter contains the number of times the backend server was transitioned to the down state following failed health checks.
Type:	Counter64
Access:	read-only
Incremented:	This counter is incremented when the backend server transitions from the up state to the down state.

Name:	bwLoadBalancerBackendServerStatusLastChange
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This date contains the last time a change in the status of this backend server occurred.
Type:	DateAndTime
Access:	read-only
Incremented:	This date is updated when the backend server state changes.

Name:	bwLoadBalancerBackendServerStatusDowntime
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This counter contains the cumulative number of seconds the backend server was in the down state.
Type:	Counter64
Access:	read-only
Incremented:	This counter is incremented while the backend server is in the down state.

Name:	bwLoadBalancerBackendServerStatusCheckStatus
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This string contains the status of the last health check to the backend server.
Type:	DisplayString
Access:	read-only
Incremented:	This string is updated after every health check.

Name:	bwLoadBalancerBackendServerStatusCheckCode
MIB:	BW-LoadBalancer.mib
Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This gauge contains the status code of the last health check to the backend server.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated after every health check.

Name:	bwLoadBalancerBackendServerStatusCheckDuration
MIB:	BW-LoadBalancer.mib

Module:	enterprises.broadsoft.broadworks.loadBalancer.backendServers
Description:	This gauge contains the number of milliseconds it took to complete the last health check to the backend server.
Type:	Gauge32
Access:	read-only
Incremented:	This gauge is updated after every health check.

11.1.2 Modified Counters

There is no impact.

11.1.3 Deleted Counters or Module

There is no impact.

11.2 Fault Management Impacts

11.2.1 New Alarms

The Load Balancer application supports the following new alarms.

Fault Name	Attributes	Value
bwPMLoadBalancerHAProxyLaunched	Problem type	Notification
	Severity range	Informational
	Subcomponent	processmonitor
	Description	This notification indicates that the Load Balancer application has started.
	Problem text	The Load Balancer application has started.
	Problem text parameters	None
	Recommendation text	None
	Recommendation text parameters	None

Fault Name	Attributes	Value
bwPMLoadBalancerHAProxyShutDown	Problem type	Notification
	Severity range	Informational
	Subcomponent	processmonitor
	Description	This notification indicates that the Load Balancer application has shut down.
	Problem text	The Load Balancer application has shut down.
	Problem text parameters	None
	Recommendation text	None

Fault Name	Attributes	Value
	Recommendation text parameters	None

Fault Name	Attributes	Value
bwPMLoadBalancerHAProxyRestarted	Problem type	Notification
	Severity range	Informational
	Subcomponent	processmonitor
	Description	This notification indicates that the Load Balancer application has restarted.
	Problem text	The Load Balancer application has restarted.
	Problem text parameters	None
	Recommendation text	None
	Recommendation text parameters	None

Fault Name	Attributes	Value
bwPMLoadBalancerHAProxyDeath	Problem type	Notification
	Severity range	Critical
	Subcomponent	Processmonitor
	Description	This notification indicates that the Load Balancer application has abnormally terminated.
	Problem text	The Load Balancer application has terminated.
	Problem text parameters	None
	Recommendation text	Make sure Load Balancer application gets restarted.
	Recommendation text parameters	None

Fault Name	Attributes	Value
bwPMLoadBalancerHAProxyOutOfMemory	Problem type	Notification
	Severity range	Critical
	Subcomponent	Processmonitor
	Description	This notification indicates that the Load Balancer application exhausted its memory allocation.
	Problem text	The Load Balancer application has exhausted its memory allocation.
	Problem text parameters	None
	Recommendation text	Make sure Load Balancer application gets restarted.

Fault Name	Attributes	Value
	Recommendation text parameters	None

The Load Balancer controller application supports the new alarms in the following table.

Fault Name	Attributes	Value
bwPMLoadBalancerHAProxyControllerLaunched	Problem type	Notification
	Severity range	Informational
	Subcomponent	processmonitor
	Description	This notification indicates that the Load Balancer application has started.
	Problem text	The Load Balancer application has started.
	Problem text parameters	None
	Recommendation text	None
	Recommendation text parameters	None

Fault Name	Attributes	Value
bwPMLoadBalancerHAProxyControllerShutDown	Problem type	Notification
	Severity range	Informational
	Subcomponent	processmonitor
	Description	This notification indicates that the Load Balancer application has shut down.
	Problem text	The Load Balancer application has shut down.
	Problem text parameters	None
	Recommendation text	None
	Recommendation text parameters	None

Fault Name	Attributes	Value
bwPMLoadBalancerHAProxyControllerRestarted	Problem type	Notification
	Severity range	Informational
	Subcomponent	processmonitor
	Description	This notification indicates that the Load Balancer controller application has restarted.
	Problem text	The Load Balancer controller application has restarted.

Fault Name	Attributes	Value
	Problem text parameters	None
	Recommendation text	None
	Recommendation text parameters	None

Fault Name	Attributes	Value
bwPMLoadBalancerHAProxyControllerDeath	Problem type	Notification
	Severity range	Critical
	Subcomponent	Processmonitor
	Description	This notification indicates that the Load Balancer controller application has abnormally terminated.
	Problem text	The Load Balancer controller application has terminated.
	Problem text parameters	None
	Recommendation text	Make sure Load Balancer controller application gets restarted.
	Recommendation text parameters	None

Fault Name	Attributes	Value
bwPMLoadBalancerHAProxyControllerOutOfMemory	Problem type	Notification
	Severity range	Critical
	Subcomponent	Processmonitor
	Description	This notification indicates that the Load Balancer utility application exhausted its memory allocation.
	Problem text	The Load Balancer controller application has exhausted its memory allocation.
	Problem text parameters	None
	Recommendation text	Make sure Load Balancer controller application gets restarted.
	Recommendation text parameters	None

11.2.2 Modified Alarms

This is not applicable.

11.3 Scripts and Tools

The bwListPortInUse script is updated to show the ports used by the Load Balancer application. These include the TCP ports used for the HTTP proxy, the TCP ports used to retrieve the statistics, and the UDP ports used to gather the logs.

11.4 EMS Integration Impacts

The *Element Management System (EMS)* panels shown in the figures in the following subsections are impacted by this feature.

11.4.1 Load Balancer

The following panels are added for the Load Balancer application that exists on the Xtended Services Platform.

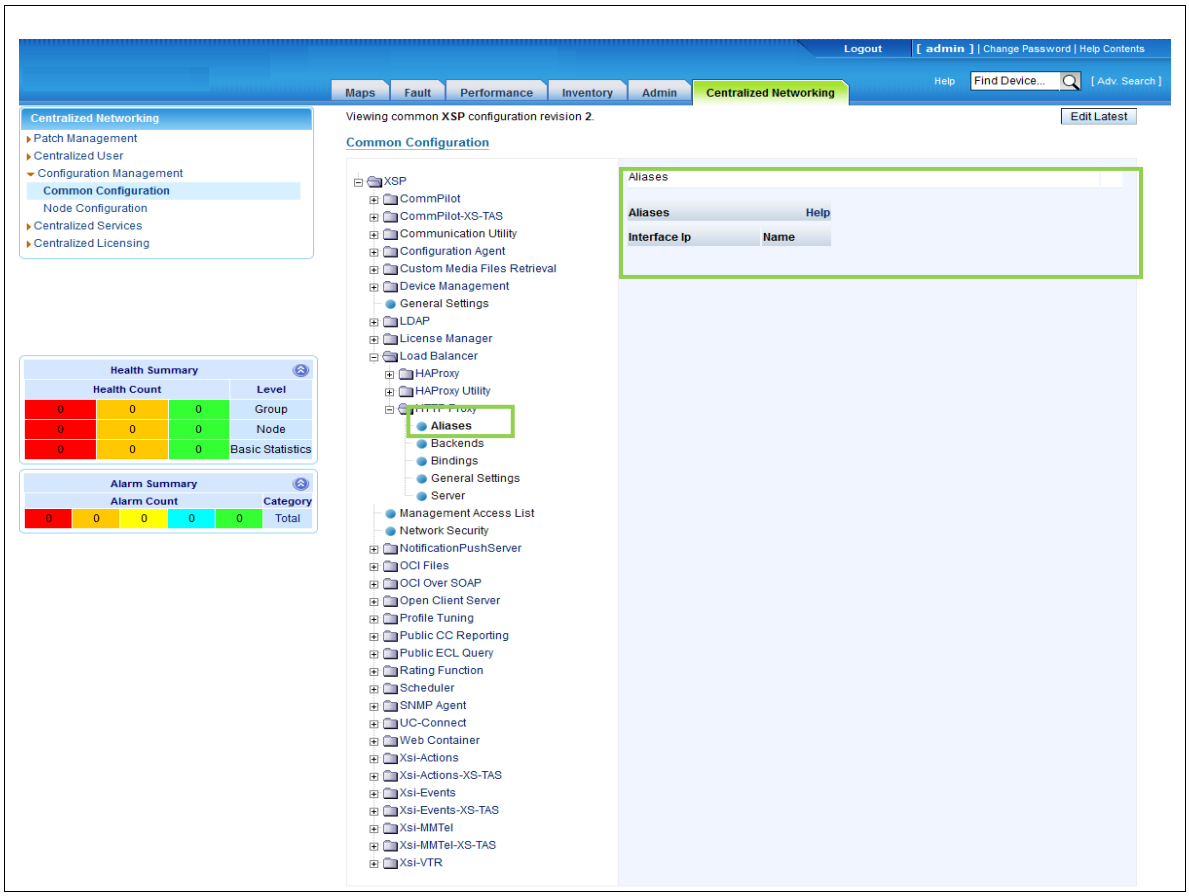


Figure 7 Load Balancer/HTTP Proxy/Aliases

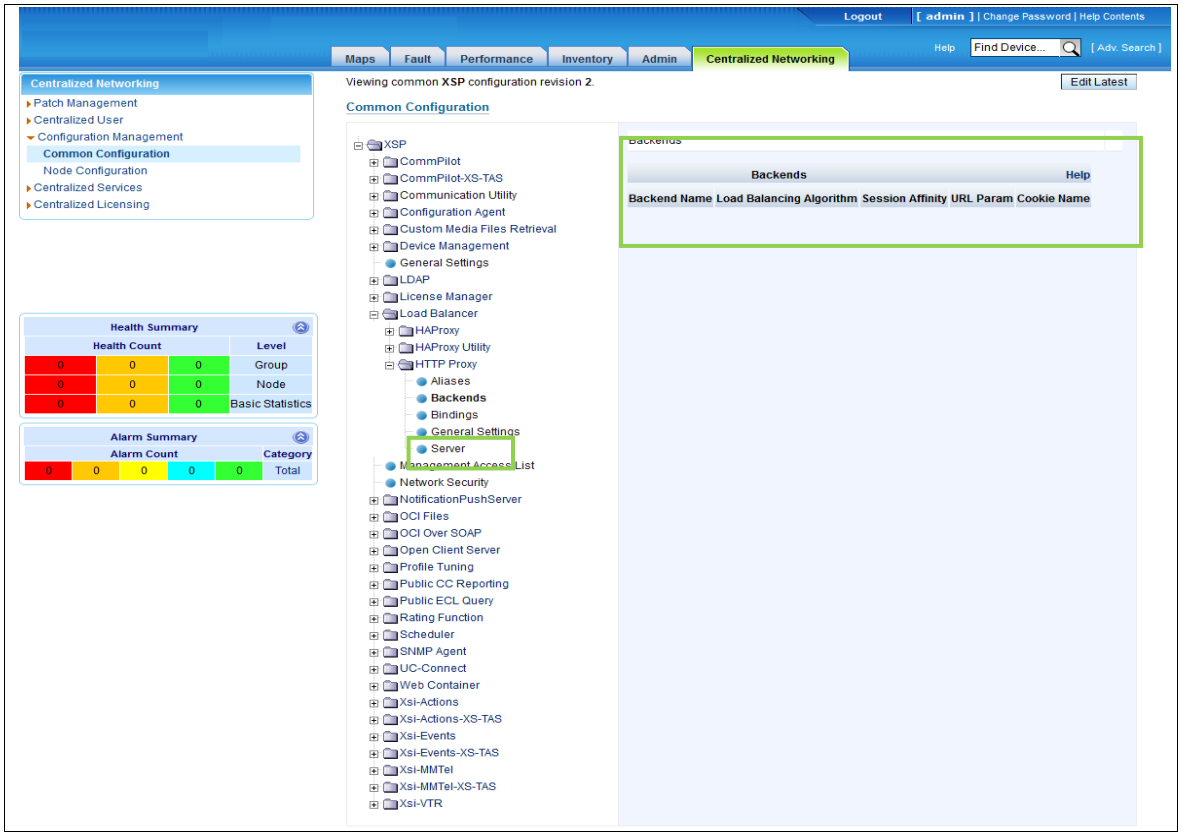


Figure 8 Load Balancer/HTTP Proxy/Backends

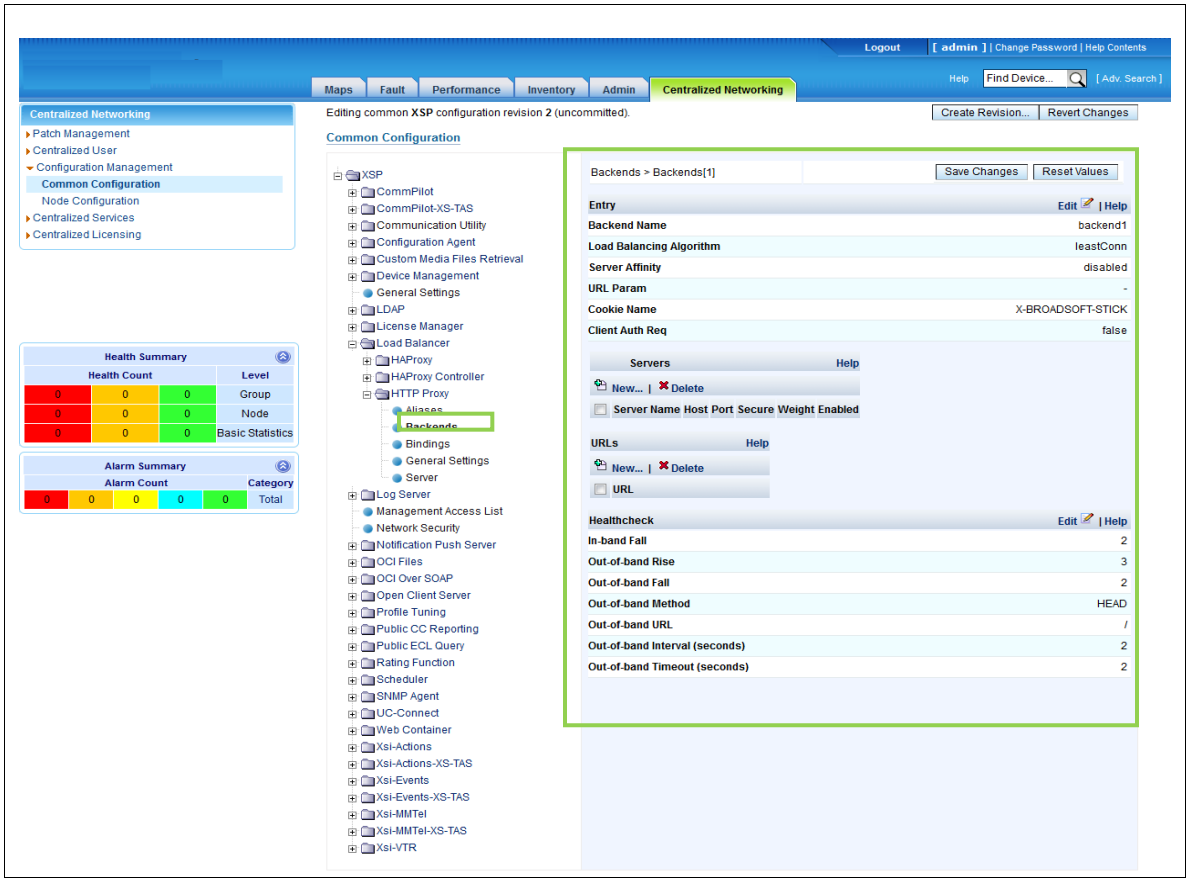


Figure 9 Load Balancer/HTTP Proxy/Backends Details

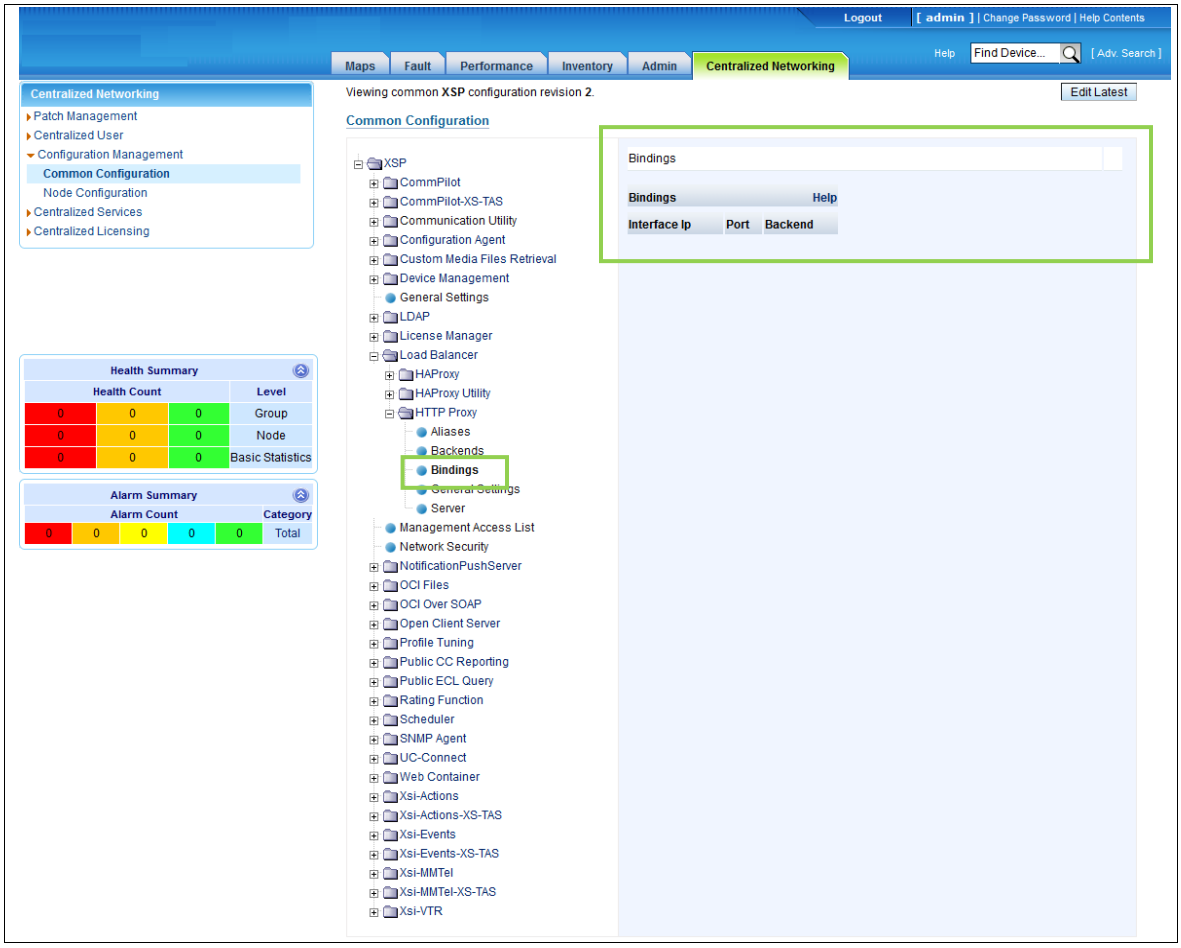


Figure 10 Load Balancer/HTTP Proxy/Bindings

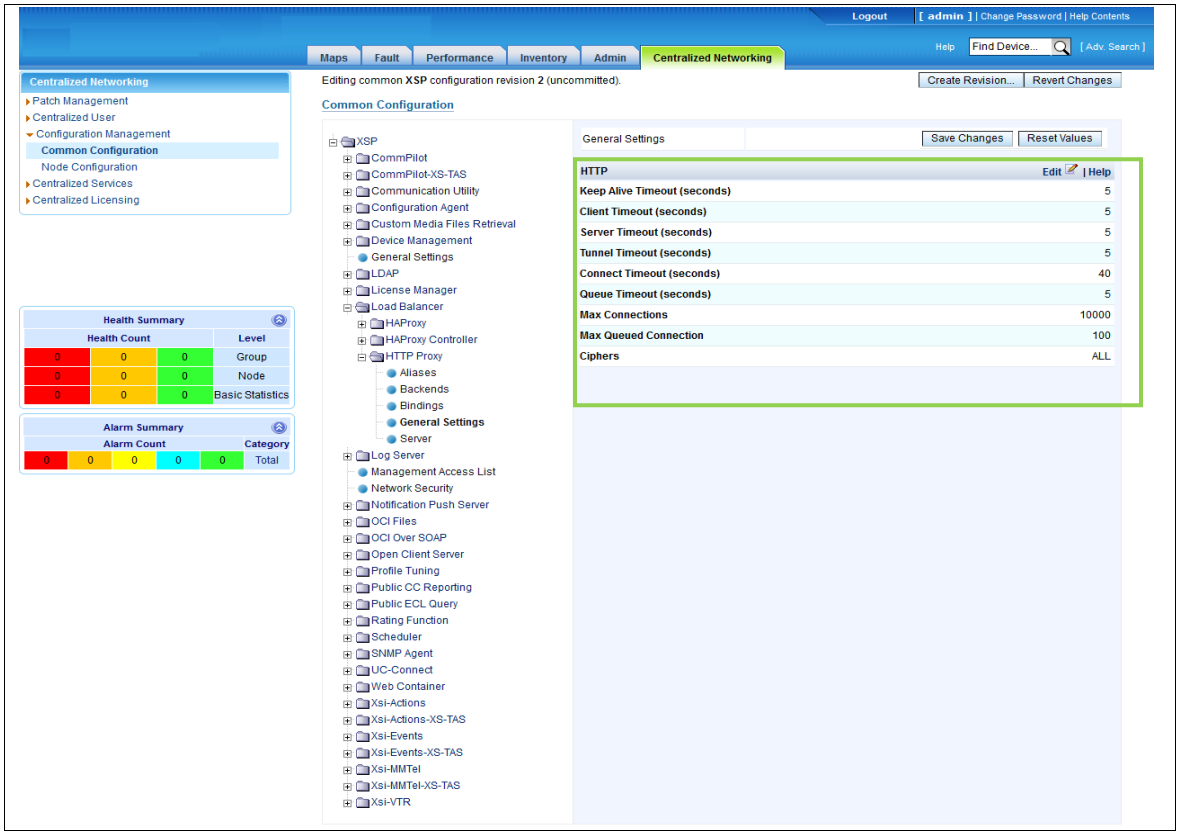


Figure 11 Load Balancer/HTTP Proxy/General Settings

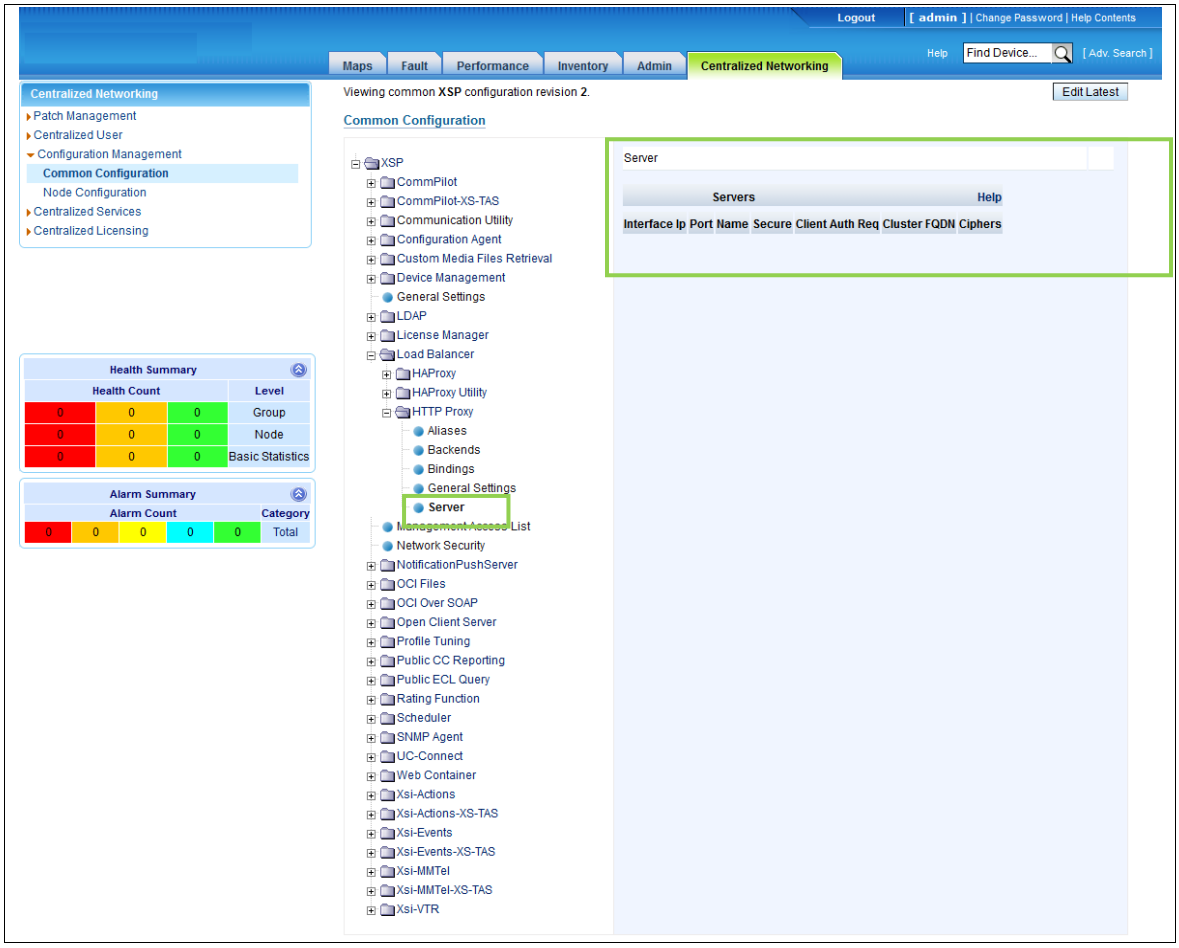


Figure 12 Load Balancer/HTTP Proxy/Server

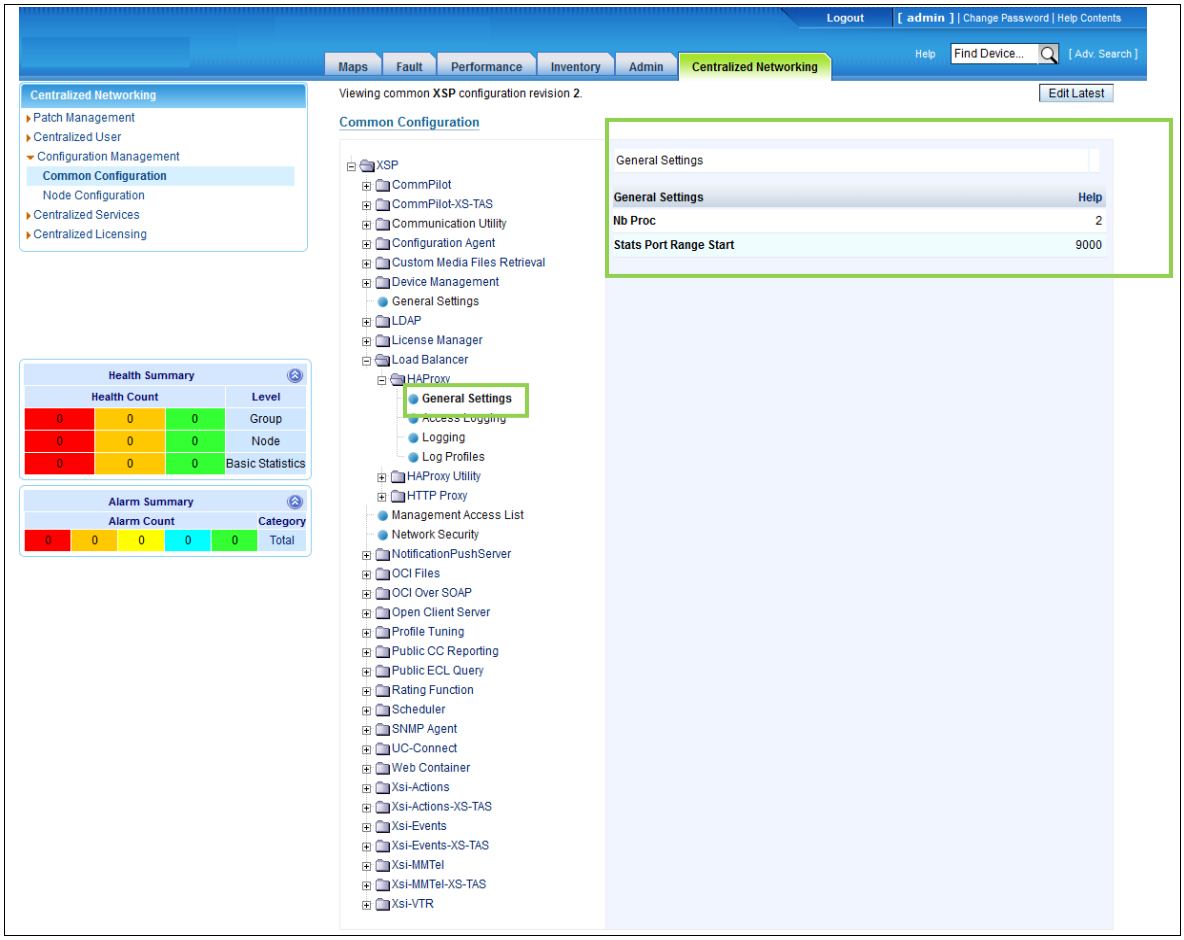


Figure 13 Load Balancer/HAProxy/General Settings

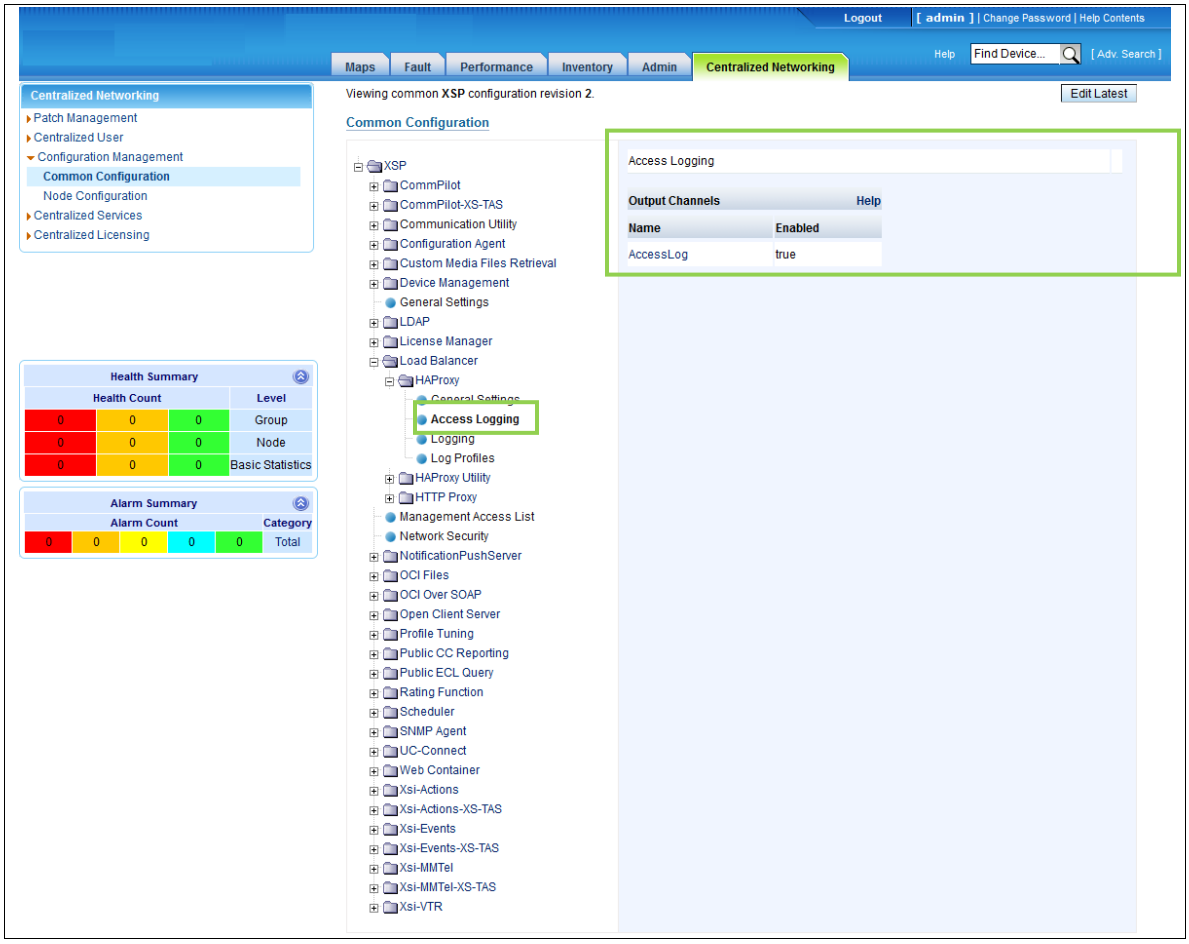


Figure 14 Load Balancer/HAProxy/Access Logging

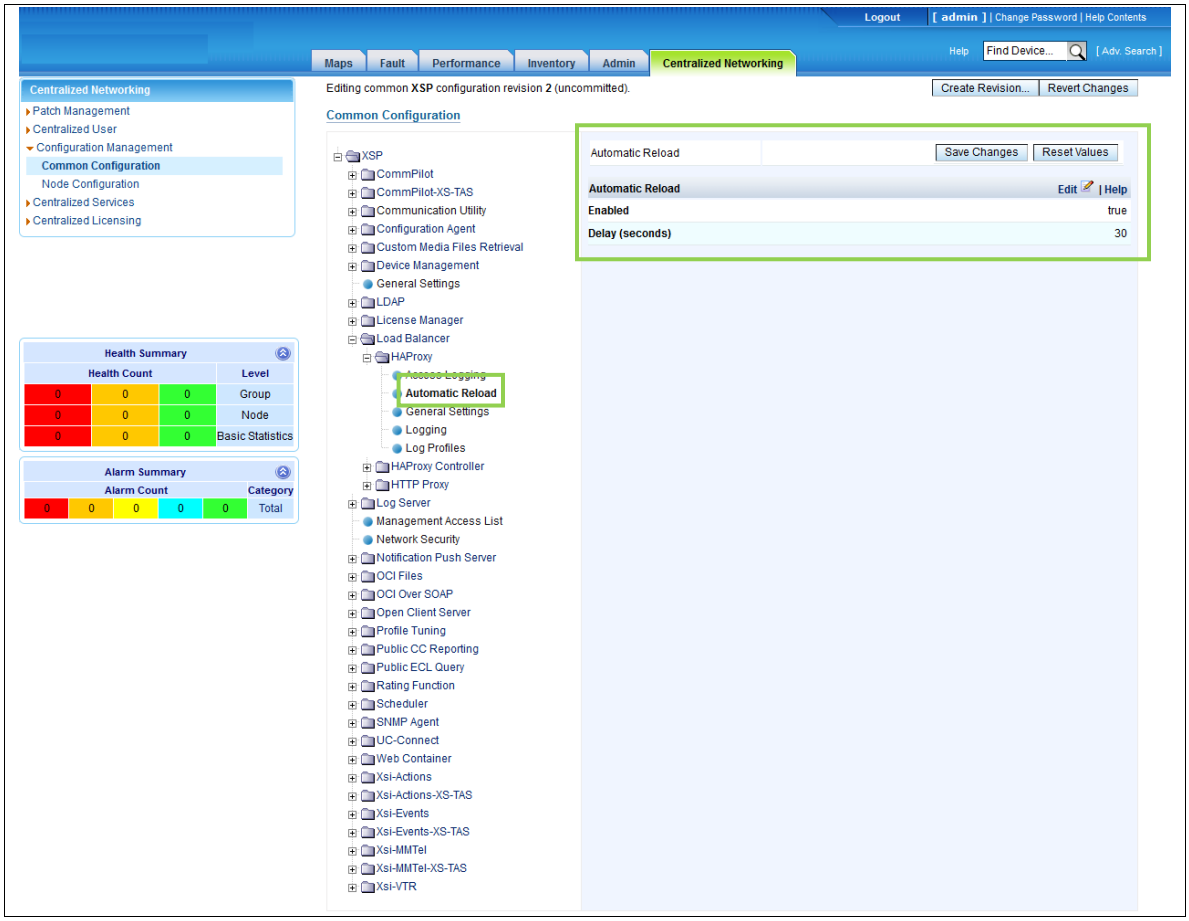


Figure 15 Load Balancer/HAProxy/Automatic Reload

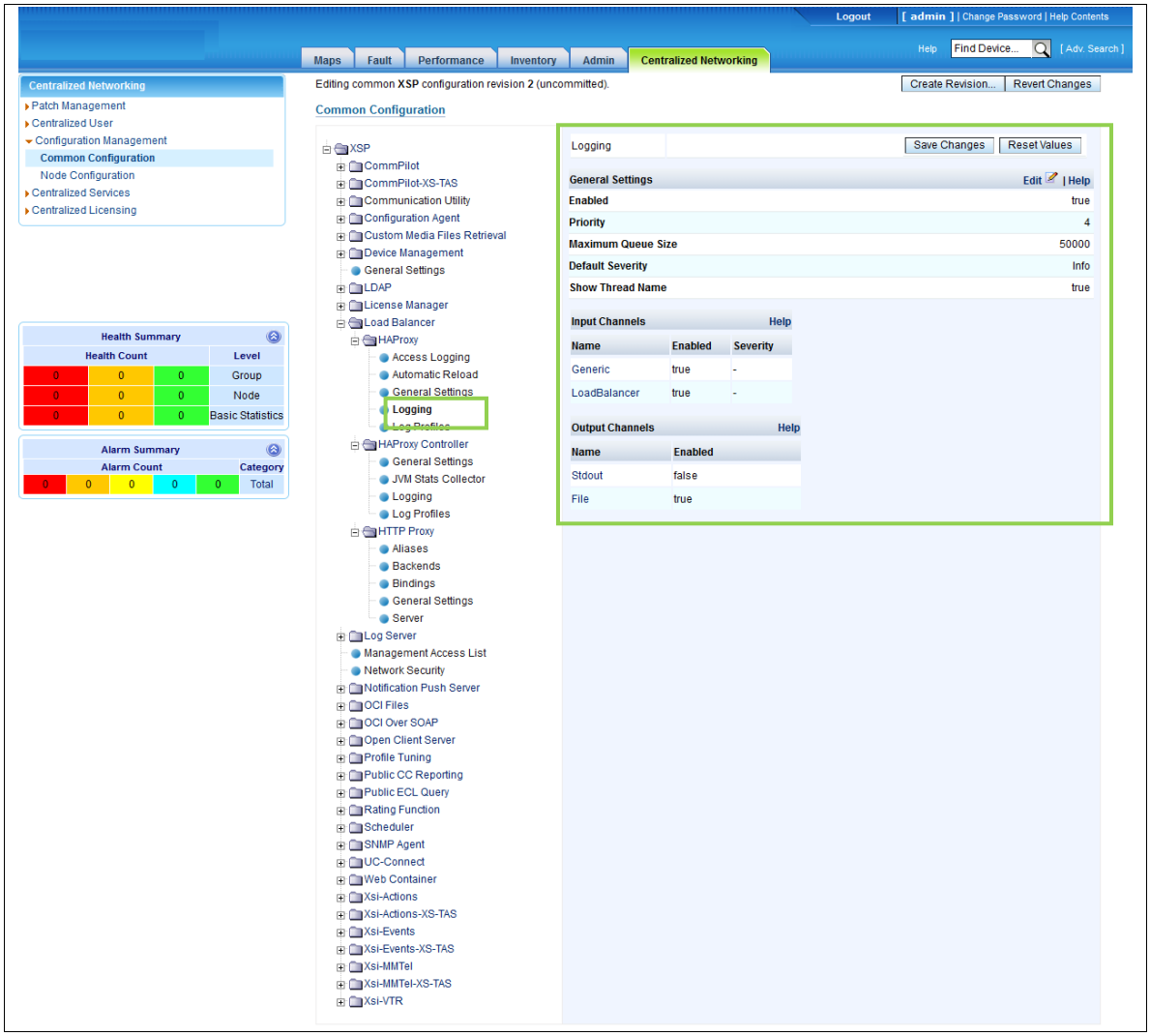


Figure 16 Load Balancer/HAProxy/Logging

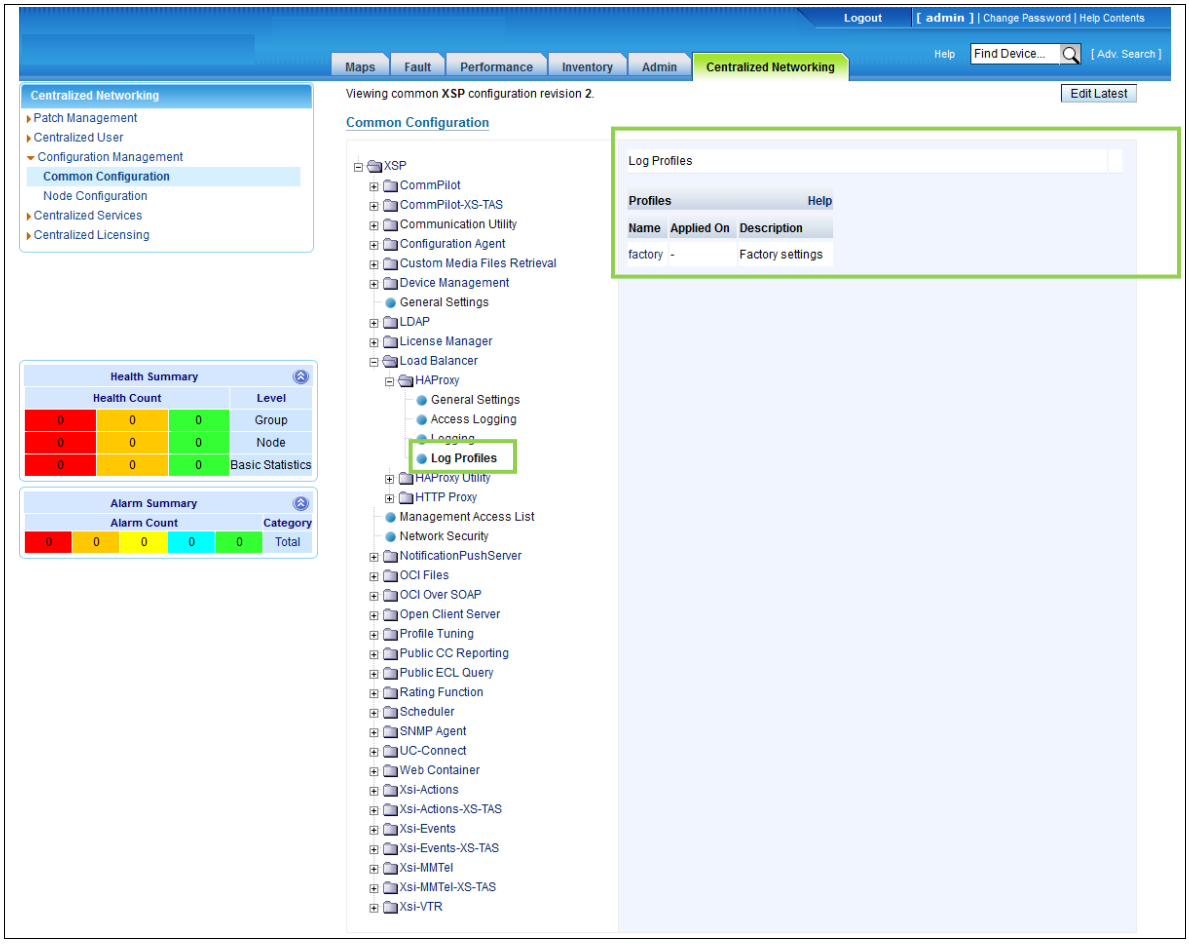


Figure 17 Load Balancer/HAProxy/Log Profiles

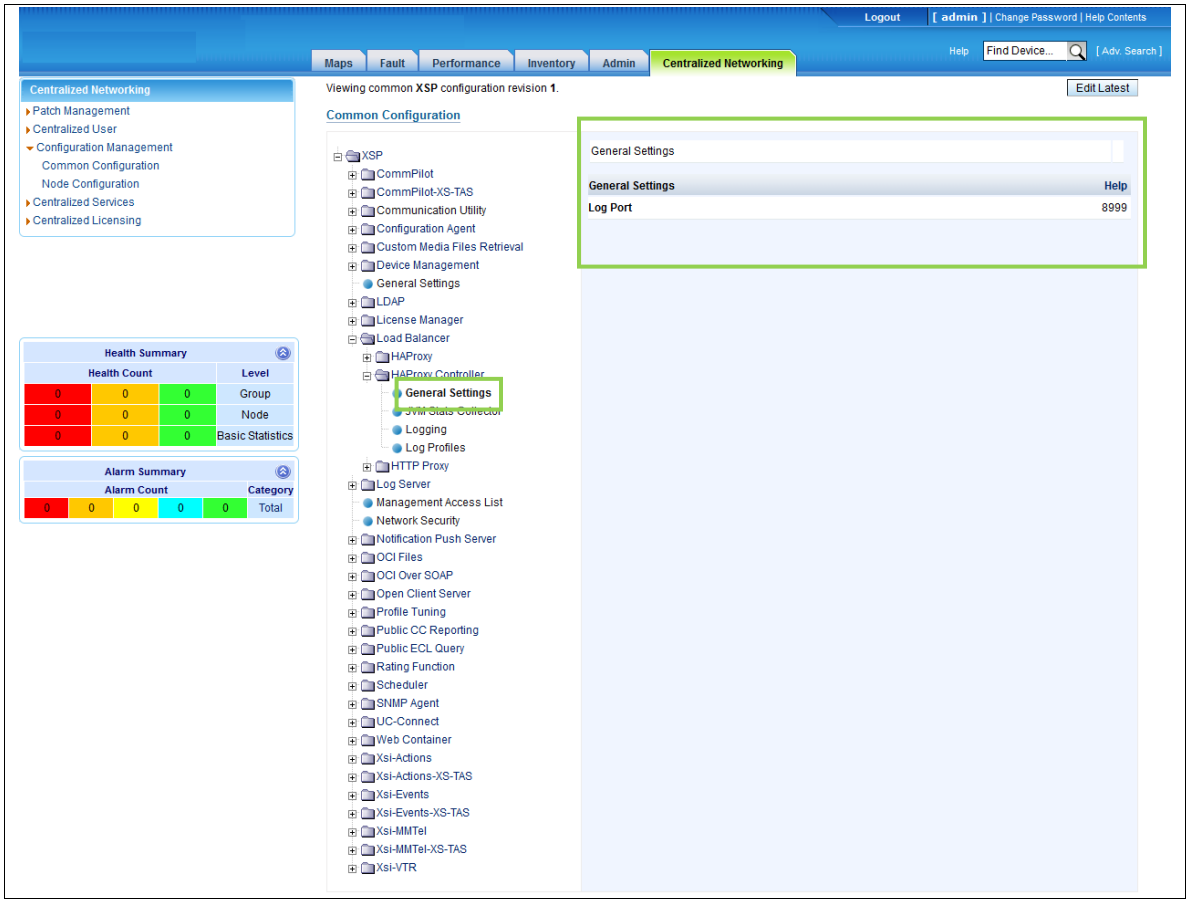


Figure 18 Load Balancer/HAProxy Controller/General Settings

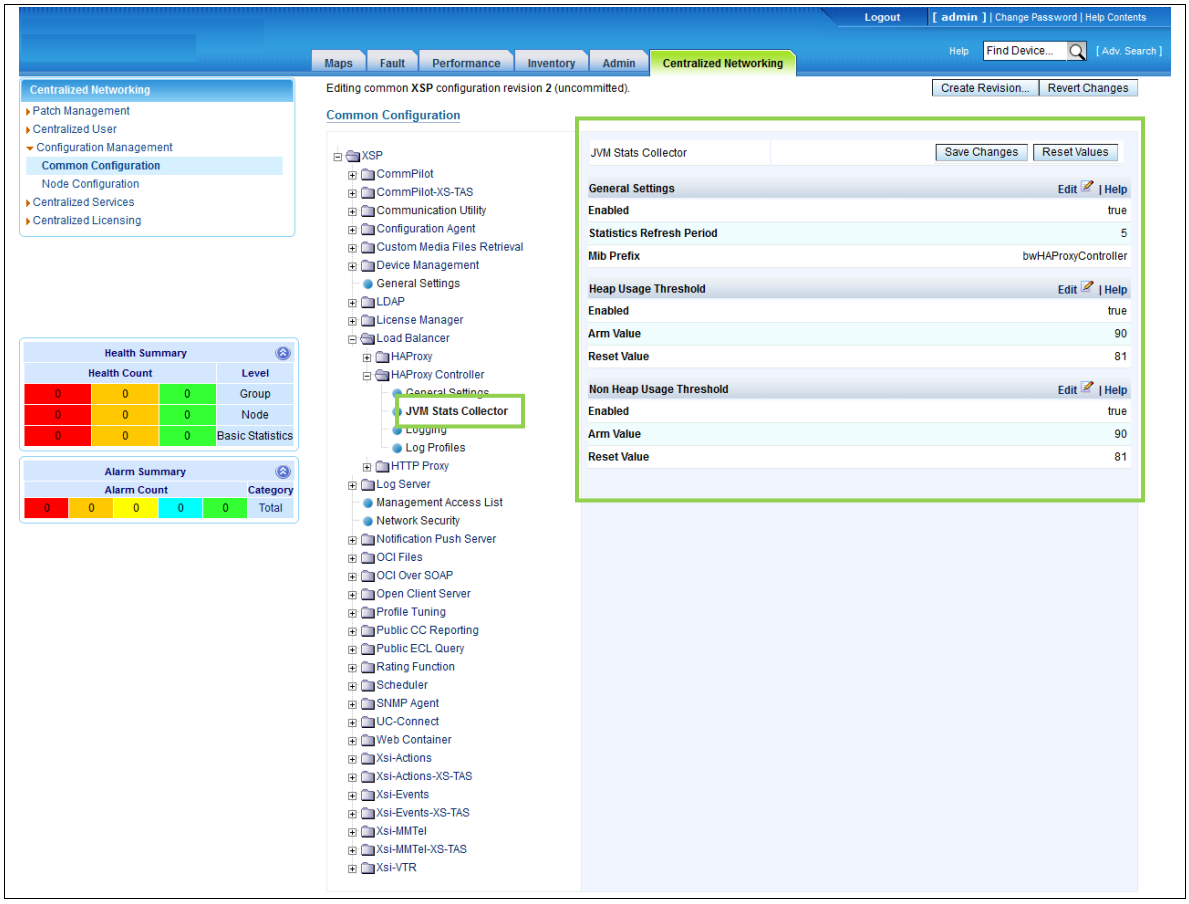


Figure 19 Load Balancer/HAProxy Controller/JVM Stats Collector

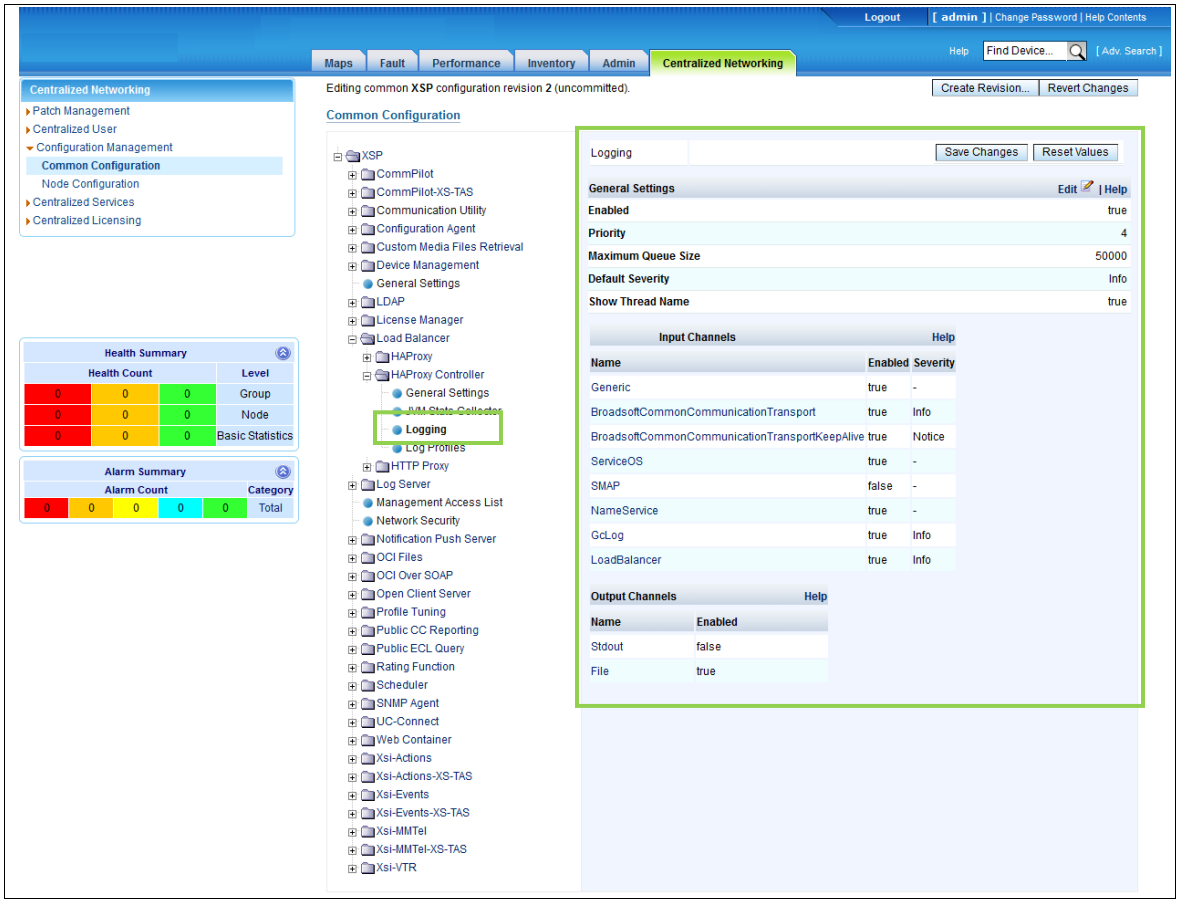


Figure 20 Load Balancer/HAProxy Controller/Logging

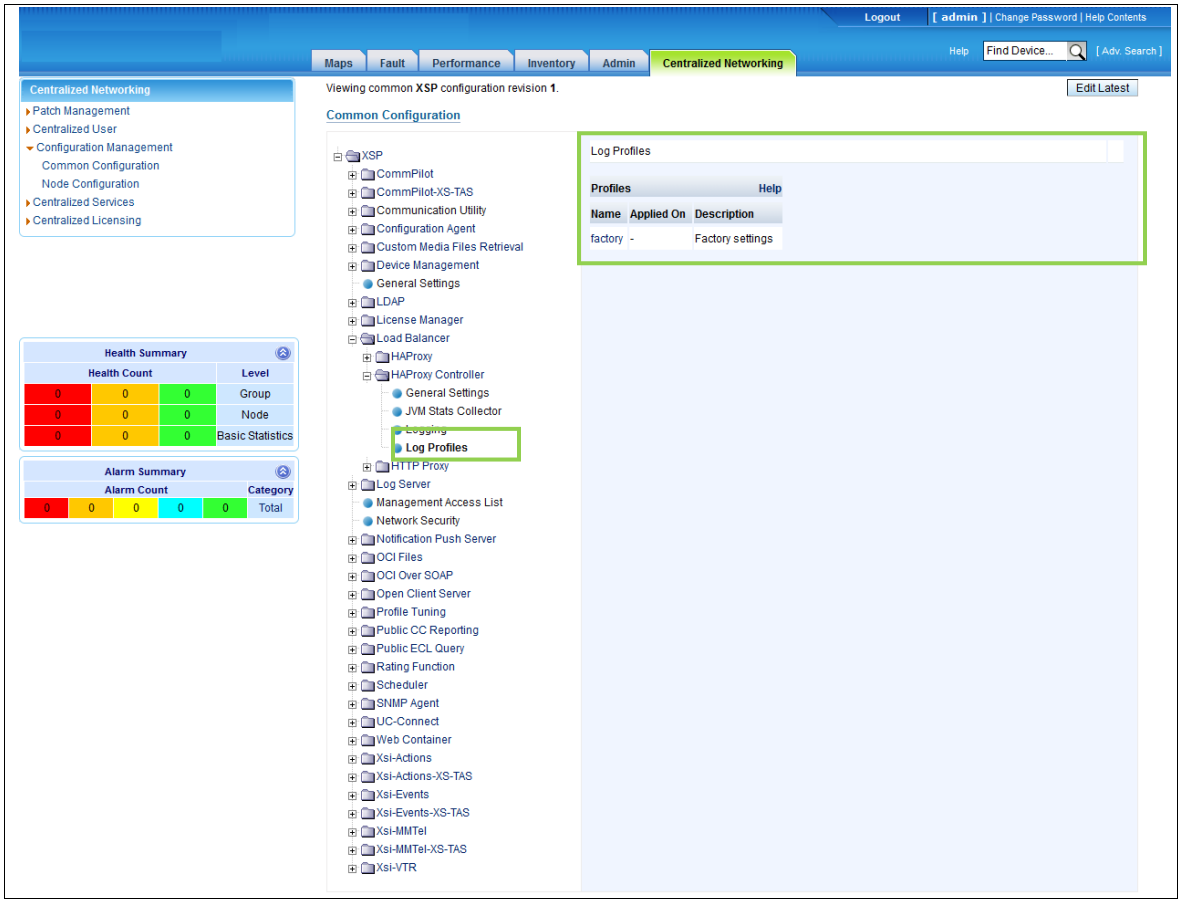


Figure 21 Load Balancer/HAProxy Controller/Log Profiles

11.4.2 Web Container

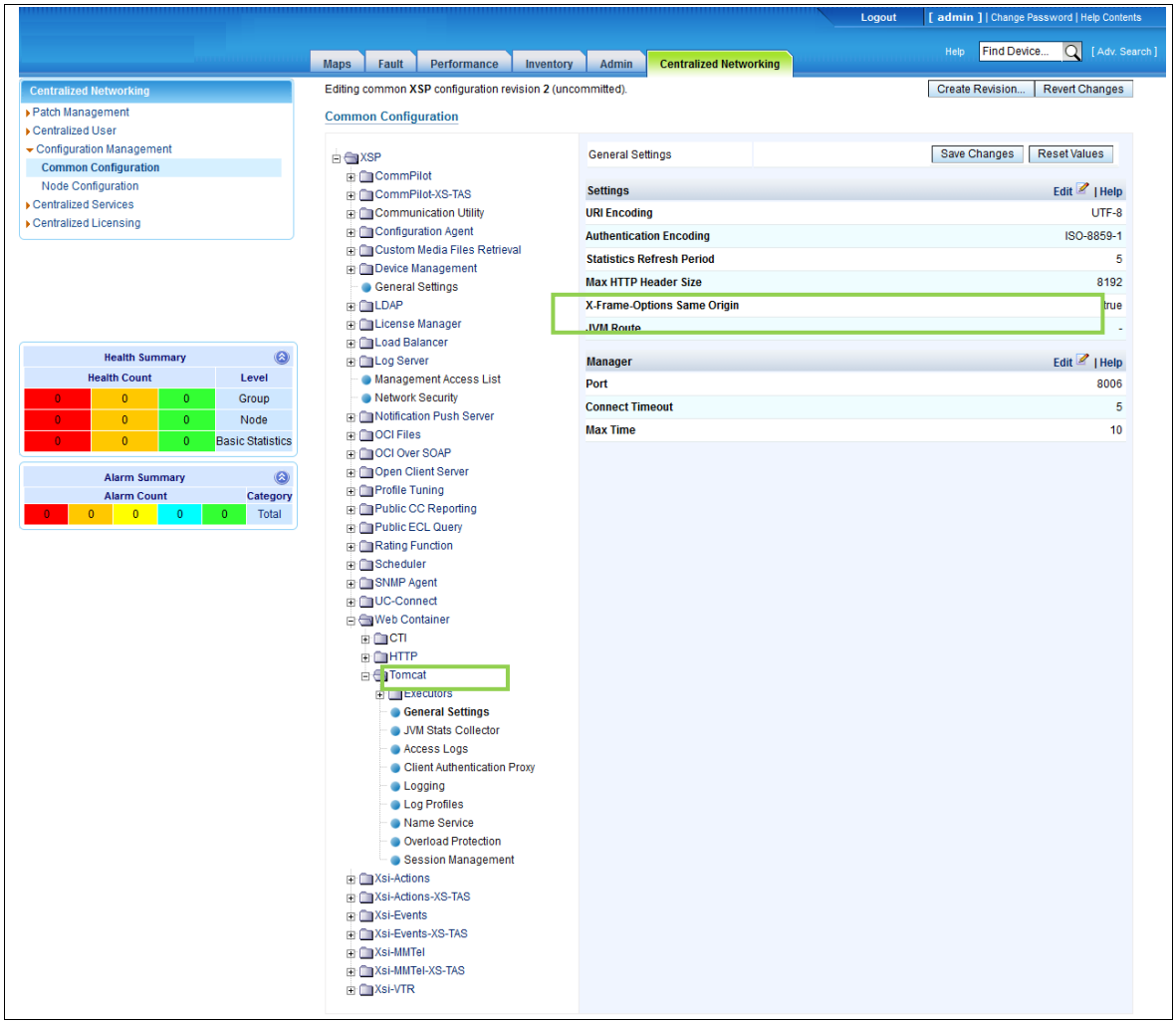


Figure 22 Web Container/Tomcat/General Settings

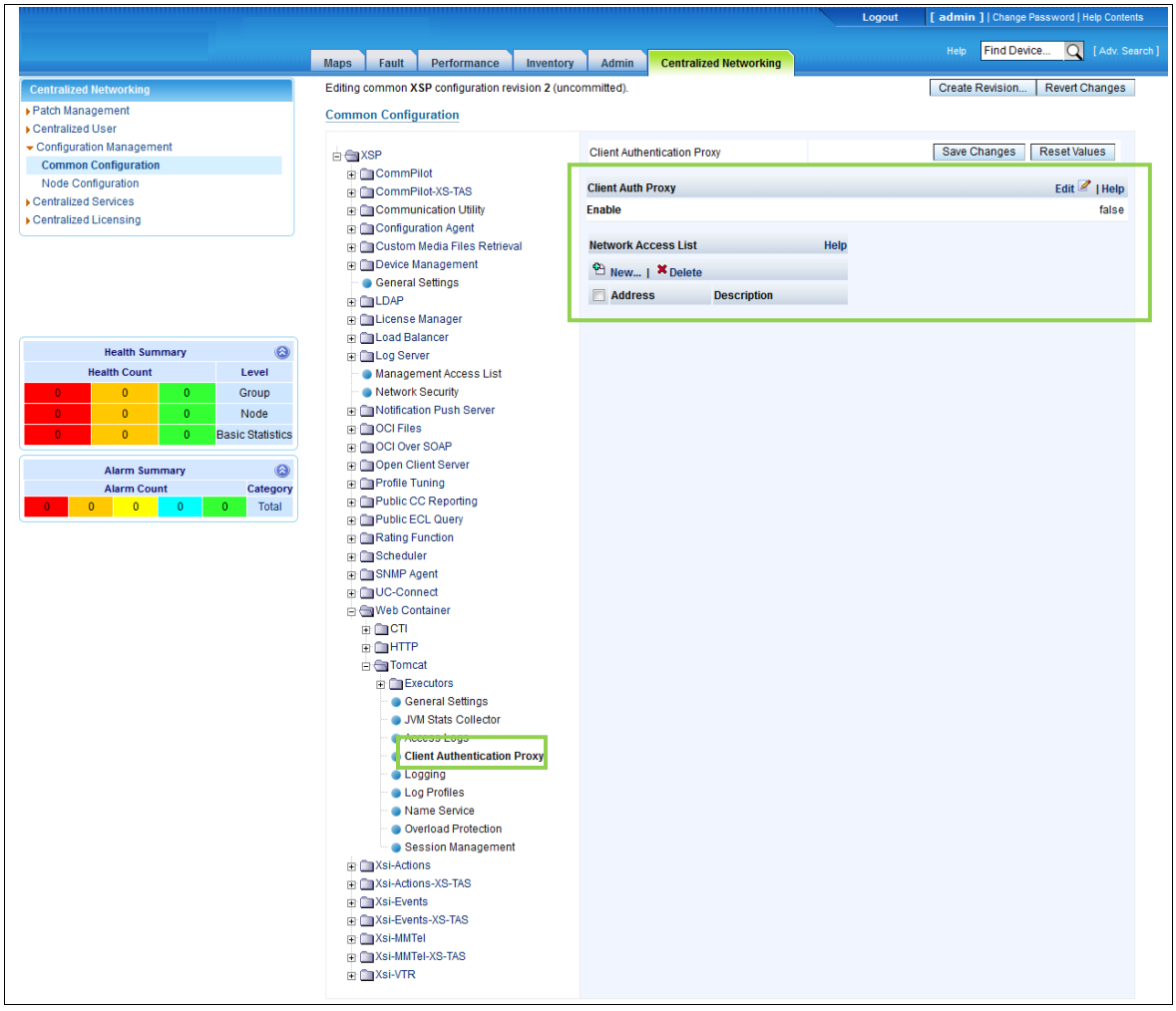


Figure 23 Web Container/Tomcat/Client Authentication Proxy

12 Execution/Call Processing Impacts

12.1 CAP Interface Impact

There is no impact.

12.2 Xtended Services Interface Impact

There is no impact.

12.3 SIP/MGCP Interface Impact

There is no impact.

13 Client Application Impacts

There is no impact.

14 Deployment/Operational Impacts

14.1 Configuration File Impacts

A configuration file is generated for *HAProxy* from the data found in the centralized configuration. It is located in the *HAProxy* application directory and it is not meant to be modifiable: */usr/local/haproxy/haproxy_base/haproxy.conf*.

14.2 Security Impacts

The Load Balancer can open TCP ports in the privileged range. Rather than run as root, the Load Balancer uses the services provided by the platform to grant access to these ports.

The SSL certificates used for its secure HTTP interfaces use the same framework as the one used by the Web Container application. As such, private keys are secured on the file system as being only readable and writable by members of the *bworks* UNIX group.

HAProxy requires that SSL certificates and keys be bundled in the same file. This bundle's access privilege is the same as the key's access privileges.

The statistics interface is unsecure and it does not use any credentials. Access control is done by opening the associated TCP ports on the local interface. Therefore, the statistics interface is not accessible from outside the system.

Several defense mechanisms are put in place to protect the system against input/output abuses.

The number of inbound connections is limited and modifiable through a configuration option. The bound interfaces are configurable, and the services and web applications accessible through each interface can be restricted with the use of bindings.

Some limits are applied to incoming HTTP requests. The maximum number of headers is limited to 101. A request containing more than that number of headers is rejected with a 400 status code response. Furthermore, request headers are limited in size by the size of the receive buffer, which is set to 16 kilobytes.

A time-out is also applied to the complete reception of an HTTP request. This time-out triggers if a complete request is not received after a set amount of time. This protects against attacks that would attempt to keep connections alive longer by sending HTTP headers slowly. The value of this time-out is equal to the keep-alive time-out of the connection.

An additional UDP port is added to the Log Server interface. It is opened on the same interface as the existing UDP port and it is opened with the same permission. The only difference is that its purpose is to receive syslog messages sent from *HAProxy* rather than binary format logs from other Cisco BroadWorks applications.

14.2.1 Security Toolkit Impact

There is no impact.

14.2.2 Application Server Default Hardening Impact

There is no impact.

14.3 Scheduled Tasks

This is not applicable.

14.4 Third-Party Software

14.4.1 Add HAProxy Version 1.5.16

The third-party software listed in the following table is added to Cisco BroadWorks.

Name	HAProxy
Description	*HAProxy* is a free, very fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications. It is particularly suited for very high traffic web sites and powers quite a number of the world's most visited ones.
Version	1.5.16
Home Page	http://www.haproxy.org/
Integration Type	Source code compiled as is.
License Type	Open Source, GPL and LGPL
License Agreement Page	http://www.haproxy.org/download/1.5/doc/LICENSE
File Names	haproxy_1.5.16.tar.gz
BroadWorks Servers	Xsp
BroadWorks Versions	22.0 and up

14.5 Server Logging Impacts

The Load Balancer generates its logs in a syslog format and sends them to a UDP socket. A controller application is used to gather these logs and write them to disk using the platform's logging infrastructure.

14.6 Client Application Impacts

There is no impact.

15 System Engineering Impacts

15.1 Processing Impacts

15.1.1 New Time-Outs

The Load Balancer uses several time-outs.

- The *connect* time-out refers to the amount of time the Load Balancer waits for a connection to be established to a backend server before aborting the attempt. Two such attempts are made, after which the request is redispatched to another server. The default value for this time-out is “5” seconds.
- The *keep-alive* time-out refers to the amount of time a session is kept in the Load Balancer, without activity. A session encompasses both a client connection and its corresponding server connection. After a session has not received any activity for the duration of the keep-alive period, the session is destroyed and both client and server connections are closed. The default value for this time-out is “5” seconds.
- The *queue* time-out refers to the amount of time a request can wait inside the Load Balancer when it is not immediately possible to establish a server connection because of resource limits. After this time-out expires, the request is dropped and a 503 error is returned to the client. The default value for this time-out is “5” seconds.

15.2 Memory Impacts

Each session established on the Load Balancer uses memory for both the client connection and the server connection. This memory footprint is about 16 KB per session. Similarly, the Load Balancer requires two file descriptors per session, one for the client connection and one for the server connection. A small number of additional file descriptors are needed for other uses.

The memory allocation is neither limited nor enforced but is nonetheless set to 70 percent as a guideline for profile tuning since the Load Balancer is generally deployed in place of the Web Container. Other profile tuning settings are also based on the values set for the Web Container.

15.3 Disk Usage Impacts

There is no impact.

15.4 Port Usage Impacts

In addition to the HTTP ports used by the proxy servers, the Load Balancer uses local TCP ports to make its statistics available. It is necessary to have one port for each process started. These ports are selected in a contiguous range and the start of this range is configurable.

An additional port is used by the Log Server to listen to UDP syslog messages sent from the Load Balancer. This new port is active on all servers on which the Log Server is deployed, even if the Load Balancer is absent.

15.5 Hardware Impacts

There is no impact.

15.6 Client Application Messaging Impacts

There is no impact.

16 Service Patch Interface Impacts

16.1 Service Patch Interface Differences

There is no impact.

16.2 Feature Activation Impacts

There is no impact.

Acronyms and Abbreviations

This section lists the acronyms and abbreviations found in this document. The acronyms and abbreviations are listed in alphabetical order along with their meanings.

AAC	Account/Authorization Code
ABNF	Augmented Backus-Naur Form
ACC	Advanced Call Control
ACD	Automatic Call Distribution
ACL	Access Control List
ACR	Anonymous Call Rejection
Admin	Administrator
AMS	Access Mediation Server
API	Application Programming Interface
AS	Application Server
AVP	Attribute Value Pair
BCCT	BroadWorks Common Communication Transport
BLF	Busy Lamp Field
BW	BroadWorks
CAP	Client Application Protocol
CBF	Communication Barring – Fixed
CCRS	Call Center Reporting Server
CDR	Call Detail Record
CDS	Call Detail Server
CFA	Call Forwarding Always
CFB	Call Forwarding Busy
CFNA	Call Forwarding No Answer
CFNR	Call Forwarding Not Reachable
CFS	Call Forwarding Selective
CLI	Command Line Interface
CLID	Calling Line ID
CNAM	Caller ID with NAMe
CORBA	Common Object Request Broker Architecture
CPL	Call Processing Language
CPU	Central Processing Unit
CRS	Call Recording Server
CS	Conferencing Server

CSCF	Call Session Control Function
CSTA	Computer Supported Telecommunications Applications
CSV	Comma Separated Value
CTI	Computer Telephony Integration
CWT	Call Waiting Tone
dBm	The power ratio in decibel (dB) of the measured power referenced to one milliwatt (mW)
dBm0	The level of a signal as specified in dBm0, is the level of that signal (in dBm) as measured at the reference point of the network.
DBS	Database Server
DGC	Distributed Group Calls
DM	Device Management
DN	Directory Number
DND	Do Not Disturb
DNS	Domain Name System
DOS	Disk Operating System
DPUBI	Directed Call Pickup with Barge-in
EMS	Element Management System
EOCP	Enhanced Outgoing Calling Plan
EV	ExtraView
FAC	Feature Access Code
FD	Feature Description
FQDN	Fully Qualified Domain Name
FR	Feature Request
FS	Functional Specification
FTP	File Transfer Protocol
HCB	Hierarchical Communication Barring
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
Hz	Hertz
ICP	Incoming Calling Plan
IMAP	Internet Message Access Protocol
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IVR	Interactive Voice Response
JVM	Java Virtual Machine
LI	Lawful Intercept

LO	Local
LPS	Local Premium Service
LSSGR	LATA Switching Systems Generic Requirements
MAC address	Media Access Control address
MB	Megabyte
MGCP	Media Gateway Control Protocol
MIB	Management Information Base
MOC	Microsoft Office Communications
MR	Market Request
MS	Media Server
NCOS	Network Class of Service
NE	Network Element
NS	Network Server
NSSync	Network Server Synchronization
OAM&P	Operations, Administration, Management, and Provisioning
OCI	Open Client Interface
OCI-C	Open Client Interface-Call Control
OCI-P	Open Client Interface-Provisioning
OCI-R	Open Client Interface-Reporting
OCP	Outgoing Calling Plan
OCS	Open Client Server
ODP	Outgoing Digit Plan
OID	Object Identifier
OOTB	Out-of-the-Blue
OS	Operating System
OSS	Operations Support System
PBX	Private Branch Exchange
PCV	P-Charging-Vector
PDF	Portable Document Format
PM	Performance Measurement
PSI	Public Service Identity
PSTN	Public Switched Telephone Network
PTT	Push To Talk
PUI	Public User Identity
RAM	Random Access Memory

RFC	Request for Comments
RTP	Real-Time Transport Protocol
SAC	Session Admission Control
SBC	Session Border Controller
SCA	Shared Call Appearance
SCA	Selective Call Acceptance
SCR	Selective Call Rejection
SDR	Session Data Replication
SIP	Session Initiation Protocol
SMAP	Software Management Application Protocol
SMDI	Simplified Message Desk Interface
SMPP	Short Message Peer-to-Peer Protocol
SMS-C	Short Message Service Center
SMTP	Simple Mail Transfer Protocol
SNI	Server Name Indication
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SP	Service Patch
SRV	Service Locator
SSH	Secure Shell
SSL	Secure Sockets Layer
TAS	Telephony Application Server
TCP/IP	Transmission Control Protocol/Internet Protocol
TDM	Time Division Multiplexing
TLS	Transport Layer Security
TO	Toll
TPS	Toll Premium Services
TXNS	Transactions
UDP	User Datagram Protocol
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VMS	Voice Mail System
VoIP	Voice Over Internet Protocol
WebDAV	Web-based Distributed Authoring and Versioning
WS	Web Server

XML	eXtensible Markup Language
XS	Execution Server
XSD	XML Schema Definition
Xsi	Xtended Services Interface
Xsp	Xtended Services Platform