broadsoft

broadsoft broadworks

Profile Server

Configuration Guide

Release 21.0

Document Version 2

9737 Washingtonian Boulevard, Suite 350
Gaithersburg, MD 20878
Tel +1 301.977.9440

WWW.BROADSOFT.COM

BroadWorks® Guide

Copyright Notice

Trademarks

Any product names mentioned in this document may be trademarks or registered trademarks of BroadSoft or their respective companies and are hereby acknowledged.

This document is printed in the United States of America.

Document Revision History

Release	Version	Reason for Change	Date	Author
14.0	1	Created document.	June 4, 2008	Yves Racine
14.0	1	Edited and published document.	September 11, 2008	Andrea Fitzwilliam
15.0	1	Updated document for Release 15.0 and published.	September 12, 2008	Andrea Fitzwilliam
16.0	1	Updated document for Release 16.0.	May 25, 2009	Yves Racine, Martin Bernier Mario Goupil
16.0	1	Edited changes and published document.	August 14, 2009	Andrea Fitzwilliam
17.0	1	Updated document for Release 17.0.	February 21, 2010	Yves Racine
17.0	1	Edited changes and published document.	April 1, 2010	Andrea Fitzwilliam
17.sp1	1	Updated document for Release 17.sp1.	October 4, 2010	Yves Racine
17.sp1	1	Edited changes and published document.	November 17, 2010	Andrea Fitzwilliam
17.sp1	2	Made minor changes.	November 25, 2010	Yves Racine
17.sp1	2	Edited and published document.	November 25, 2010	Jessica Boyle
17.sp1	3	Updated PS application list.	January 13, 2011	Mario Goupil
17.sp1	3	Edited changes and published document.	August 5, 2011	Jessica Boyle
18.0	1	Updated document for release 18.0.	September 26, 2011	Yves Racine
18.0	1	Edited changes and published document.	November 11, 2011	Jessica Boyle
19.0	1	Updated document for Release 19.0.	October 3, 2012	Yves Racine
19.0	1	Updated document to reflect changes introduced by EV 147586 WebContainer Platform Enhancements in Release 19.0.	October 24, 2012	Eric Ross Martin Bernier
19.0	1	Edited changes and published document.	November 13, 2012	Patricia Renaud
20.0	1	Updated document to reflect changes following Device Management Device File Security activity and Device Management Profile Server Directory Enhancement.	September 9, 2013	Pierre Drapeau
20.0	1	Edited changes and published document.	October 4, 2013	Joan Renaud
21.0	1	Updated document for Release 21.0.	September 24, 2014	Yves Racine
21.0	1	Edited changes and updated copyright notice.	October 14, 2014	Joan Renaud
21.0	2	Updated section *8.8 Redundancy* to include all necessary applications for PR-47417.	November 12, 2015	Goska Auerbach
21.0	2	Edited changes and published document.	April 11, 2016	Jessica Boyle

Table of Contents

Table of Figures

1 Summary of Changes

The following subsection identifies the changes to this document for each document version.

1.1 Changes for Release 21.0, Document Version 2

This version of the document includes the following changes:

- Updated section *8.8 Redundancy* to include all necessary applications for PR-47417.

1.2 Changes for Release 21.0, Document Version 1

The document has been updated to reflect the content of Release 21.0.

1.3 Changes for Release 20.0, Document Version 1

This version of the document includes the following changes:

- Added the following sections with the Profile Server Directory enhancements:
  - *8.8 Redundancy*
  - *8.11.1.6 Extended File System*
  - *8.11.1.7 File Replication*
- Added section *8.11.1.5 Encryption* with the Device File Security activity for file encryption.

1.4 Changes for Release 19.0, Document Version 1

This version of the document includes the following changes:

- Updated the list of applications available on the Profile Server.
- Added the following sections to this document to reflect changes introduced by feature EV147586 WebContainer Platform Enhancements:
  - *7.2.1.1 WebContainer Software Components*
  - *7.2.1.2 WebContainer Metric Groups*
  - *8.2 Profile Tuning*
  - *8.9.5 Executors*
  - *8.9.7 Name Service*
  - *8.4 Configuration of WebContainer*
  - *8.5 Management of WebContainer*

  Updated the following sections:
  - *8.9.4 Overflow Protection Settings*
  - *8.7.7 HTTP Bindings*
  - *8.11.1.5 Overflow Protection Settings*
  - *8.12.1.4 Overflow Protection Settings*
  - *8.13.1.4 Overflow Protection Settings*

- *8.14.3.4 Overflow Protection Settings*

1.5 Changes for Release 18.0, Document Version 1

This version of the document includes the following change:

- Updated the list of applications available on the Profile Server.

1.6 Changes for Release 17.sp1, Document Version 3

This version of the document includes the following change:

- Updated the list of applications available on the Profile Server.

1.7 Changes for Release 17.sp1, Document Version 2

This version of the document includes the following change:

- Made minor changes to the document.

1.8 Changes for Release 17.sp1, Document Version 1

The document has been modified to include new applications for Home Subscriber Server (HSS) deployment.

1.9 Changes for Release 16.0, Document Version 1

This version of the document includes the following changes:

- Changed section *8.7.5 Manage SSL Certificates* to add a reference to the sslRemove and sslExport commands and introduce the new certificate chain parameter of the sslUpdate command.
- Revised section on changing the command line interface (CLI) root of the File Repository application to point to *Applications/BroadworksFileRepos_16.0*.

1.10 Changes for Release 15.0, Document Version 1

There were no changes to this document for Release 15.0.

1.11 Changes for Release 14.0, Document Version 1

The document was created for this release.

2 Introduction

The Profile Server (PS) is a general-purpose application-hosting server that is part of the internal service provider network and is not directly accessible from the public network; only the Application Servers, Xtended Services Platforms, the Execution Server (XS), and the Messaging Server (UMS) need to communicate with the Profile Server.

The Profile Server can be configured as a centralized file repository in the context of Device Management, as the profile provisioning engine in the IP Multimedia Subsystem (IMS) architecture, or as the repository function for the Instant Messaging and Presence.

The Profile Server supports the Hypertext Transfer Protocol (HTTP) (with or without WebDAV extensions), which the Application Servers, the Xtended Services Platforms, and the Messaging Server use to push and retrieve files to/from the Profile Server.

Similar to the Xtended Services Platform, the Profile Server runs Apache and Tomcat; however, it also packages a WebDAV component running in Tomcat.

As the Profile Server is based on the Xtended Services Platform, it can be set up to support one or more HTTP servers, each one either handling secure HTTP traffic or not handling secure HTTP traffic, and listening on a configurable port number.

Tomcat is the centerpiece of the Profile Server. The WebDAV support is implemented through servlets running within Tomcat. Tomcat provides all the required threading and session models to support multiple WebDAV operations simultaneously.

Even if the Profile Server is based on the Xtended Services Platform, it does not allow installing, activating, and deploying other applications.

2.1 Deployment Models

The Profile Server is deployed in farm architecture and each Profile Server uses rsync or HTTP to exchange files between other Profile Servers in the farm.

Figure 1 shows the major software components of the Profile Server and how they relate one another.

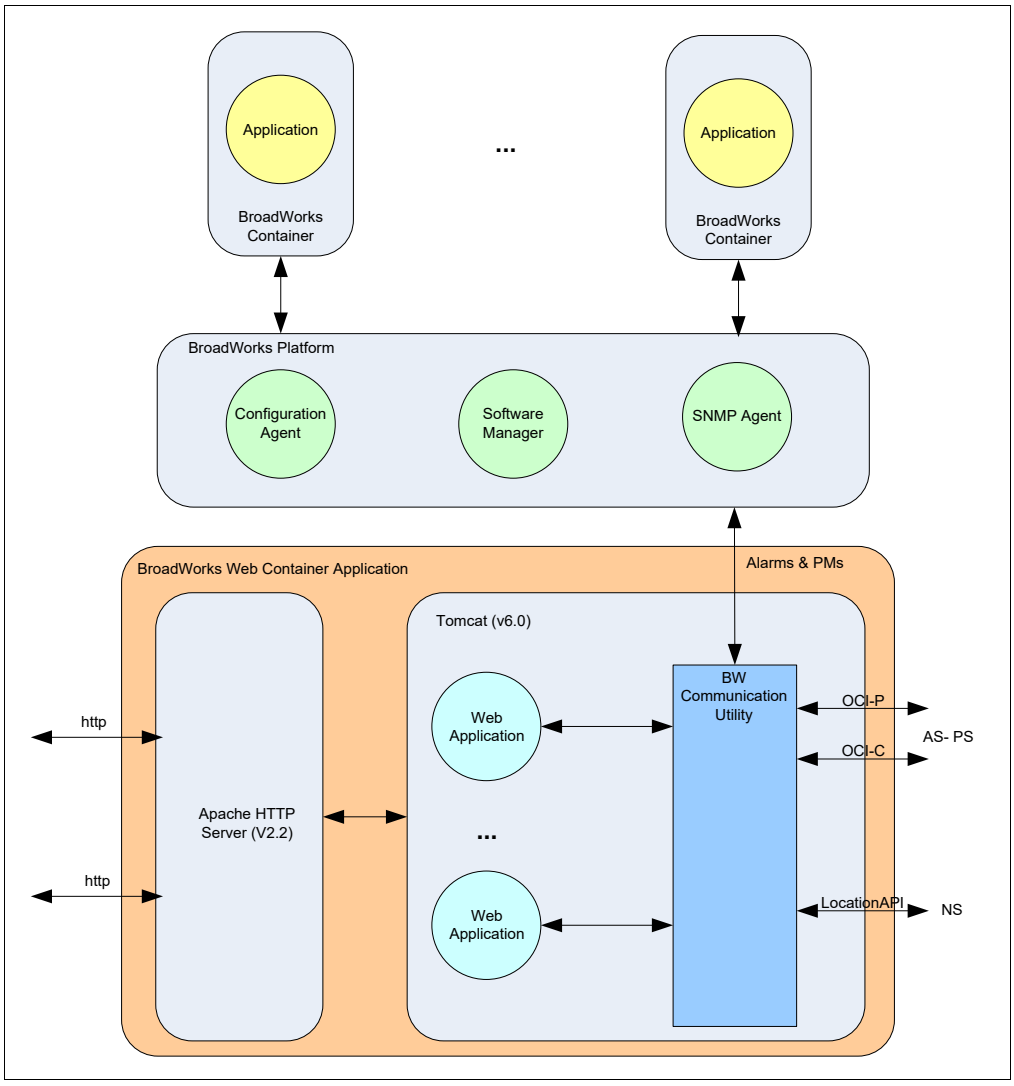

Figure 1 Components Overview

This guide provides general product description and deployment information of the Profile Server.

3 Getting Started

Execute the following steps to start a Profile Server.

1) Obtain a license from BroadSoft. For more information, see section *5 Licensing*.
2) Install the Profile Server from the binary image. For more information, see the *BroadWorks Software Management Guide* [1].
3) Configure the required applications depending on the context.
4) Start the Profile Server.

4 Software Distribution

The Profile Server is distributed using binary images for Linux. The image files are named using the usual format:

- *PS_Rel_21.0_1.yyy.Linux-x86_64.bin*

Where yyy is a unique identifier for the Profile Server Release 21.0.

5 Licensing

The Profile Server platform has its own license. This license can contain multiple licenses to allow the running of BroadWorks applications.

The Profile Server does not start if it is not properly licensed.

6 Installation, Upgrade, and Patching

The installation procedure for the Profile Server is similar to the procedure used for existing BroadWorks servers. For detailed information and procedures on installing the Profile Server, see the *BroadWorks Software Management Guide* [1].

The upgrade and patching process for the Profile Server works the same way as it does for other BroadWorks servers. However, the BroadWorks and web applications hosted by the Profile Server are handled differently from the Profile Server itself. The following subsection provides more information.

6.1 Server Upgrade

The upgrade of the Profile Server is similar to that of the other BroadWorks servers except for the activation. Compared to the other servers where the activation is performed in two steps (setting the target version and resetting), activating the Profile Server is performed in a single step, that is, setting the active software version. As soon as the set command is executed and accepted, the server is stopped and its version is switched to the specified version. Finally, the server is started with the specified version. The following example shows the set command.

```
PS_CLI/Maintenance/ManagedObjects> help set
The SET command is used to modify some managed objects-related
attributes.

Set
    [<forceOption>, Choice = {force}]
    <option>, Choice = {activeSoftwareVersion}
        activeSoftwareVersion:
            <type>, Choice = {server}
                server:
                    <identity>, String {1 to 80 characters}
                    <version>, String {1 to 80 characters}

PS_CLI/Maintenance/ManagedObjects>

PS_CLI/Maintenance/ManagedObjects> set activeSoftwareVersion server PS
21.sp1_1.269

+++ WARNING +++ WARNING +++ WARNING +++
This command will change the active software version of PS to
21.sp1_1.269 NOTE that this action will cause downtime.
 Continue?

Please confirm (Yes, Y, No, N): y
Adding
/var/broadworks/logs/installation/setactiveserver.PS.21.sp1_1.269.2010060
7_144942.log to logPrinter
BroadWorks SW Manager activating PS server version 21.sp1_1.269…
…
```

NOTE: Server upgrade shall be performed during the maintenance window.

6.2 Applications Installation, Upgrade, and Patching

The Profile Server includes/embeds a fixed list of BroadWorks applications. All applications are automatically installed during the Profile Server installation process.

The following table provides information regarding packaged applications.

Name	Type	Release	Upgrade Mode	Default State
BroadworksFileRepos	BW	21.0	Automatic	Deployed
CCReporting	BW	21.0	Automatic	Installed
CCReportingDbManagement	BW	21.0	Automatic	Installed
CCReportingRepository	BW	21.0	Automatic	Installed
DBSObserver	BW	21.0	Automatic	Installed
DeviceManagementFiles	BW	21.0	Automatic	Installed
ECLQuery	BW	21.0	Automatic	Installed
ECLReportingRepository	BW	21.0	Automatic	Installed
EnhancedCallLogsDbManagement	BW	21.0	Automatic	Installed
LogRepository	BW	21.0	Automatic	Active
MediaFiles	BW	21.0	Automatic	Installed
MeetMeConfRepository	BW	21.0	Automatic	Installed
MessageArchive	BW	21.0	Automatic	Installed
OpenClientServer	BW	21.0	Automatic	Installed
Provisioning	BW	21.0	Automatic	Installed
WebContainer	BW	21.0	Automatic	Deployed

All applications are automatically upgraded by the BroadWorks upgrade process. The only non-automated task is the decision to activate/deactivate or deploy/undeploy the application, which is still made by the operator.

7 Profile Server Components

The Profile Server hosts several active components that provide internal and external services.

7.1 Core Daemons

The components described in this section are BroadWorks server daemons that are started by the operating system's *initd* service when the server boots. They are only stopped when the operating system stops. As such, these daemons are started when the operating system enters run level 3, 4, or 5 (that is, normal operation). They are terminated when the operating system leaves these levels (for example, shutdown).

Manual control of the daemon's life cycle is neither required nor expected but can be accomplished using the commands specified in the following sections.

7.1.1 Software Manager

The Software Manager is part of the core components of the Profile Server. It is responsible for the patching of the BroadWorks applications and platform. It is also responsible for the installation, activation, and deployment of BroadWorks applications and web applications. The Software Manager is a critical component that must run for an operator to perform management and maintenance of the BroadWorks server.

7.1.1.1 Life Cycle Control

While it is not recommended during normal operation, it is possible to restart the Software Manager during the maintenance window. To stop and start it, the following commands can be used.

```
bwadmin@mtl64lin12 $ stopswman.pl

Stopping SW Manager version: 108659
bwadmin@mtl64lin12 $ startswman.pl

Starting SW Manager version: 108659
bwadmin@mtl64lin12 $
```

7.1.1.2 Logs

The Software Manager logs are located in directory */var/broadworks/logs/swmanager*.

7.1.2 SNMP Agent

The Simple Network Management Protocol (SNMP) agent is a core component of the Profile Server. It is not started and stopped with BroadWorks. It is running all the time. The SNMP agent exposes the application counters and gauges to the management system. When a new application (BroadWorks or web) is installed on the Profile Server, its counters and gauges are automatically added to those exposed by the SNMP agent. On the other end, when an application is uninstalled, its counters and gauges are removed.

7.1.2.1 Life Cycle Control

While it is not recommended during normal operation, it is possible to restart the SNMP agent during the maintenance window. The life cycle of the SNMP agent is controlled with the snmpdctl script located in */usr/local/broadworks/bw_base/bin*. The script supports the following options:

- start: Starts the SNMP agent
- stop: Stops the SNMP agent
- status: Shows the current SNMP agent's state (*inService* or *dead*)
- restart: Stops and restarts the SNMP agent

Following is an example of the usage.

```
bwadmin@mtl64lin12 $ snmpdctl
Usage: snmpdctl {start|stop|restart|status}
bwadmin@mtl64lin12 $ snmpdctl stop
Shutting down bwsnmpd
bwadmin@mtl64lin12 $ snmpdctl start
Starting bwsnmpd:       [ok]
bwadmin@mtl64lin12 $
bwadmin@mtl64lin12 $ snmpdctl status
bwsnmpd: inService
```

NOTE: The statistics collected by the SNMP agent are lost when the SNMP agent is stopped.

7.1.2.2 Logs

The SNMP agent logs are located in directory */var/broadworks/logs/snmp*.

7.1.3 License Manager

The License Manager (LicenseManager) is a core component of the Profile Server. It runs all the time, regardless of whether BroadWorks is started or stopped. The License Manager is started initially during the installation process but more generally, the UNIX *initd* starts and stops it at boot-time and shutdown.

The License Manager is responsible for managing the license files locally, getting the license files from the Network Function Manager (NFM) server, and requesting license permission usage to the Network Function Manager as well.

The License Manager must be running for the applications to start. If the License Manager is down or is stopped and one attempts to start BroadWorks (startbw), the applications does not start.

7.1.3.1 Life Cycle Control

While it is not recommended during normal operation, it is possible to restart the License Manager during the maintenance window. The life cycle of the License Manager is controlled with the lmdctl script located in */usr/local/broadworks/bw_base/bin*. The script supports the following options:

- start: Starts the License Manager.
- stop: Stops the License Manager.
- status: Shows the current License Manager's state (*inService* or *dead*).
- restart: Stops and restarts the License Manager.

Following is an example of the usage.

```
bwadmin@mtl64lin12 $ lmdctl
Usage: lmdctl {start|stop|restart|status}
bwadmin@mtl64lin12 $ lmdctl stop
Shutting down lmd...            [ok]
bwadmin@mtl64lin12 $ lmdctl start
Starting lmd...                 [ok]
bwadmin@mtl64lin12 $
bwadmin@mtl64lin12 $ lmdctl status
lmd: inService
```

7.1.3.2 Logs

The License Manager logs are located in directory */var/broadworks/logs/lmd*.

7.1.4 Configuration Agent

The Configuration agent is responsible for managing the server's configuration. It hosts the configuration data and is interfaced by other BroadWorks components that need to read and/or write configuration data. The Configuration agent is a critical component that must run for other components to operate properly. It provides local services to other server components and to external services accessed by the BroadWorks Element Management System (EMS).

7.1.4.1 Life Cycle Control

It is not recommended to stop the Configuration agent during normal operation. The Configuration agent is used by many platform components and applications and when stopped, these components and applications may misbehave. When required, it can be stopped during the maintenance window using the configdctl script provided in */usr/local/broadworks/bw_base/bin*. The script supports the following options:

- start: Starts the Configuration agent.
- stop: Stops the Configuration agent if BroadWorks applications are not running.
- stopnow: Stops the Configuration agent even if BroadWorks applications are running.
- status: Shows the current Configuration agent's state (*inService* or *dead*).
- restart: Stops and restarts the Configuration agent.
- validate: Checks if the *config.xml* file is valid.

Following is an example of the usage.

```
bwadmin@mtl64lin12 $ configdctl
Usage: configdctl {start | stop | stopnow | restart | status | validate}
bwadmin@mtl64lin12 $ configdctl stop
Shutting down configd...     [ok]
bwadmin@mtl64lin12 $ configdctl stopnow
Shutting down configd...     [ok]
bwadmin@mtl64lin12 $ configdctl start
Starting configd...          [ok]
bwadmin@mtl64lin12 $ configdctl status
configd: inService
bwadmin@mtl64lin12 $ configdctl validate
Validating ...  [ok]
```

7.1.4.2 Logs

The Configuration agent issues logs to the */var/broadworks/logs/config* directory.

7.1.5 Platform Processes

On the Profile Server, the showrun command always displays two categories of processes, that is, the application and the platform processes. Following is an example of the showrun command output.

```
bwadmin@mtl64lin12.mtl.broadsoft.com$ showrun

Currently running BroadWorks Application processes:

  apache process monitor (pid=2490)
  apache (pid=2728)
  tomcat process monitor (pid=2471)
  tomcat (pid=2524)
Currently running BroadWorks Platform processes:

  BroadWorks Configuration Agent process monitor (pid=28412)
  BroadWorks Configuration Agent (pid=28434)
  BroadWorks License Manager process monitor (pid=27496)
  BroadWorks License Manager (pid=27517)
  BroadWorks Software Manager (pid=27035)
  BroadWorks SNMP Agent process monitor (pid=28578)
  BroadWorks SNMP Agent (pid=28600)

bwadmin@mtl64lin12.mtl.broadsoft.com$
```

The platform processes must always be running for the BroadWorks applications to run properly. The healthmon command monitors these processes and reports any missing platform processes.

7.2 Controllable Applications

The components described in this section are BroadWorks services that can be started and stopped by the operator. Deployed applications typically start when BroadWorks starts and stop when BroadWorks stops.

7.2.1 WebContainer

The Profile Server deploys a WebContainer application to host the web applications. The WebContainer is composed of the Apache HTTP server and the Apache Tomcat Application Server.

The WebContainer application is like any other automatic BroadWorks applications, that is, it can be activated, deployed, undeployed, and deactivated.

7.2.1.1 WebContainer Software Components

Figure 2 fragments the Web Container into software components pertinent for the metrics it provides. This representation is reused in subsequent diagrams that explain the key metrics in detail. This section provides only a summary of the metric groups. For a complete description, see section *8.5.1 Metric Groups*.

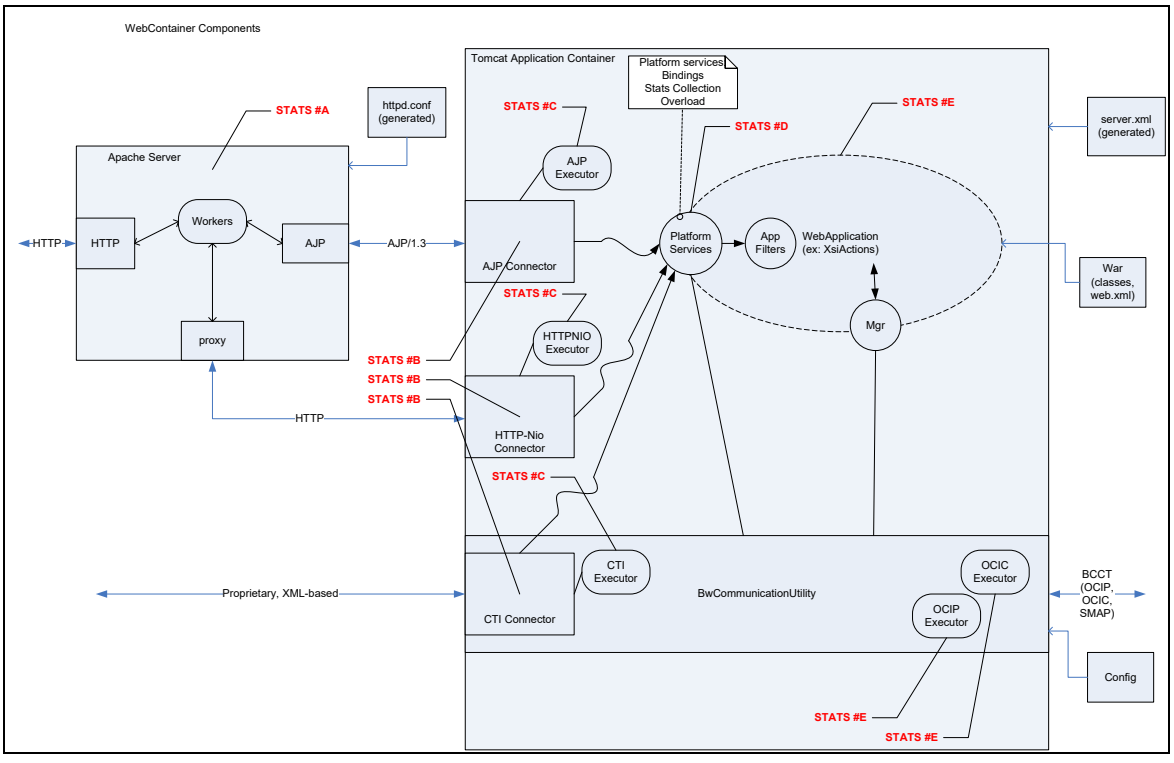

Figure 2 WebContainer Assembly

The BWCommunicationUtility provides common platform-level services that webapps can use to interact with BroadWorks services. It provides a simplified API that abstracts details of the underlying communication infrastructure. Each webapp uses a dedicated instance of the BwCommunicationMgr. The webapp can also instantiate and configure request processing filters that are provided with the BWCommunicationUtility.

The Apache Server vectors most of the incoming HTTP traffic towards Tomcat via the optimized AJP link. Incoming HTTP traffic for the /com.broadsoft.async/* url is split out towards Tomcat's specialized HttpNio connector. The HttpNio connector is used for webapps that require use of the non-blocking Comet interface. The CTIConnector is a proprietary connector that allows use of the webapps via an xml-based protocol used for CTI integration with third party systems.

7.2.1.2 WebContainer Metric Groups

As indicated in *Figure 2*, performance metrics are collected at various points within the WebContainer.

1) Apache Server metrics (Stats #A): These metrics provide server state information (busy/idle workers, uptime) and throughput information (requests served, bytes served, bytes/sec, bytes/request).
2) Connector processing metrics (Stats #B): These metrics are reported per connector type and provide total processing time (cumulative) and maximum processing time (single request).
3) Executor processing metrics (Stats #C and #E): There metrics are reported per executor type and provide measurements about queue delay and task processing time.
4) Platform Services metrics (Stats #D): These filters provide global and webapp-level metrics about overload protection, overhead processing time, and bindings.

7.2.1.3 Logs

The Apache server used by the Profile Server issues logs to the */varbroadworks/logs/apache* directory. The general settings for Apache allow configuring the log level. For full details, see section *8.4.1.1 Apache*.

The Apache Tomcat application container used by the Profile Server issues logs in the */varbroadworks/logs/tomcat* directory. This log contains the container internal logs and possibly the exceptions it encounters while executing the various web applications.

The general settings for Apache allow configuring the log level. For full details, see section *8.4.1.2 Tomcat*.

8 Configuration and Management

8.1 Configuration Modes

The Profile Server supports the following configuration modes:

- Stand-alone: Configuration is locally managed through the CLI application.
- Managed: Configuration is remotely managed through the EMS.

In Managed mode, no local configuration changes are allowed. All configuration management operations are done using the EMS. Configuration is downloaded in bulk from the EMS.

The Profile Server is installed in Stand-alone mode by default. The mode is only controllable by the EMS. The EMS transitions the Profile Server to Managed mode when directed to do so by the operator. When placed in Managed mode, the EMS assumes control of the server's configuration. The same configuration settings are available in both modes.

For Managed mode operations, see the *BroadWorks Element Management System Administration Guide* [2].

Not all the configurations of a node have been integrated in the Configuration agent. Some platform components still rely on configuration files. The following are always managed locally:

- Maintenance task configuration
- Software Manager configuration

8.2 Profile Tuning

The profile tuning provides a list of pre-defined profile definitions based on a set of identified deployment models. This flexibility eases the tailoring of a server to meet its expected usage.

The Profile Server renders a CLI level for the profile tuning under the *PS_CLI/System/ProfileTuning/GeneralSettings* level.

The profile tuning has the following characteristics:

- This functionality is optional. That is, a server functions normally without an assigned profile.
- The configuration action for a selected profile is immediate: when an administrator selects a profile, the system immediately sets the parameters with the values fetched from the profile.
- Only one profile is active at any given time. That is, the profiles are not stackable.
- The selected profile can be changed at any time. This action is immediate and supersedes all previously tailored values.
- The selected profile is reloadable at any time. Such action may be used to restore a server to the initial profile values (thus eliminating manual customization for the specific parameters included in the reloaded profile).
- When a profile includes webapp-centric data, the profile values apply upon deploying a webapp even if the webapp was not present at the time of selecting a profile.
- This is the only mechanism available for tuning the Java Garbage Collection (GC) parameters.

- It is not possible to revert or undo a profile. The CLI only indicates the profile that is currently applied. To go back to a known profile, clear the existing profile and re-apply the desired profile.

8.2.1 Profile Characterization

The following table captures the targeted characteristics when identifying the needs for a profile. In an effort to ease the documentation, the table includes only groups of parameters and targeted characteristics rather than a list of specific parameters and values.

Profile Type			**Non-Real-Time**	**Real-Time**
Server Type	**Parameter Group**	**Input parameters**	**Targeted Content**	
Profile Server	Garbage Collection (Tomcat)	Memory CPU Counts	Disable concurrent GC. Use values similar to existing defaults.	Enables concurrent GC (parallel GC threads at 100% of CPU Counts). Same value for heap min and heap max. Fixed new size generation at 100% overall heap size.
	Apache	None	Use values similar to existing defaults.	Targeted parameters: ▪ max clients and associated server limit ▪ Listen Backlog ▪ Timeout ▪ KeepAliveTimeout
	Tomcat	None	Use values similar to existing defaults.	Targeted parameters: ▪ Global session timeout ▪ Per webapp session timeout ▪ Max threads per connector type

Pre-defined Profile Servers – Targeted Characteristics

Based on these characteristics, the following profiles are available:

- PS non real-time
- PS real-time

A default profile that corresponds to the factory config settings is also available.

8.2.2 CLI Context

Following is an example of setting the profile tuning to a real-time profile.

```
PS_CLI/System/ProfileTuning/GeneralSettings> help set
This command is used to modify profile tuning general settings.

Parameters description:
attribute        : The name of an attribute to modify.
profileTuningName: The name of the selected profile tuning.

======================================================================
set
    <attribute>, Multiple Choice = {profileTuningName}
        <profileTuningName>, Choice = {default, realTime, nonRealTime}

PS_CLI/System/ProfileTuning/GeneralSettings>
```

```
PS_CLI/System/ProfileTuning/GeneralSettings> set profileTuningName
nonRealTime
...Done

PS_CLI/System/ProfileTuning/GeneralSettings> get
  profileTuningName = nonRealTime
```

8.3 WebApplication Threading

Processing of HTTP requests within web applications is performed on threads provided by the underlying platform components (WebContainer). Web Applications can have internal threads as needed but the main processing threads are those provided by the platform. The threading service is provided by the various "Executors" provided at the platform level as illustrated in *Figure 2 WebContainer Assembly*.

A distinct executor is available at each input/output location. Each executor can be individually configured for the needs of the interface it serves. The WebContainer manages the following external interfaces: AJP, HTTPNIO, and CTI. The following sections provide specific details about these executors.

8.3.1 General Concept

An executor is a means of bounding and managing resources consumed when executing a collection of tasks. It is composed of a thread pool used to execute required tasks and a queue where tasks wait to be executed. To address the level of flexibility expected from the Profile Server, the following configuration parameters are supported for each executor:

- **minPoolSize**: defined as the minimum number of live threads (busy or idle) kept in the pool at any time (excluding startup transients).
- **maxPoolSize**: defined as the maximum number of allowed live threads (busy and idle) in the pool.
- **keepAliveTime**: defined as the amount of idle time before an excess thread is terminated.
- **queueCapacity**: defined as the maximum number of tasks that can be queued while waiting for other tasks to complete their execution. The capacity can be unlimited (left unspecified), disabled (0 capacity), or fixed.

Based on these parameters, the following approach is used when processing tasks (for example, processing an incoming http request):

- The actual pool size is automatically adjusted based on the **minPoolSize** and the **maxPoolSize**. When a new task is submitted and fewer than **minPoolSize** threads are running, the system allocates a new thread to handle the request, even if other worker threads are idle. If there are more than **minPoolSize** but less than **maxPoolSize** threads running, the system allocates a new thread only if the queue is full or disabled (as expressed by the actual queue size).
- If the pool currently has more than **minPoolSize** threads, excess threads will be terminated if they have been idle for more than **keepAliveTime**. This provides a means of reducing resource consumption when the pool is not actively being used. If the pool becomes active later, new threads will be allocated. Note that the keep-alive policy applies only when there are more than **minPoolSize** threads.

As a result of this behavior, the system rejects a request when the queue is full and the actual pool size equals maxPoolSize.

The queue capacity can be left unspecified, meaning unlimited queuing. In that case, the queuing capacity is implicitly limited by the total memory capacity available to the WebContainer. The queue capacity can be set to "0", effectively disabling the queuing function (the queue can be thought of as always being full). In that case, new tasks are immediately rejected if no thread is available in the pool and the pool size has already reached its maxPoolSize.

8.3.2 Typical Configurations

Although providing a great deal of configuration flexibility, executors are typically configured following two operational models:

- Fixed threading: The thread pool is fixed for optimum processing throughput, waiting is acceptable for momentary excess demand [minPoolSize=maxPoolSize, queueCapacity > 1]
- Adjustable threading: The number of threads is adjusted with demand, waiting is not acceptable [minPoolSize < maxPoolSize, queueCapacity=0]

8.3.2.1 Configuration Through the CLI

Following is an example of configuring an Open Client Interface-Call Control (OCI-C) executor (for both queue and thread pool).

```
PS_CLI/Applications/WebContainer/Tomcat/Executors/AJP/Queue> help set
The command is used to modify Queue settings.

Parameters description:
attribute: The name of an attribute to modify.
capacity : This parameter specifies the maximum number of requests that can
be
           queued while waiting for other tasks to complete during execution.

======================================================================
set
    <attribute>, Multiple Choice = {capacity}
        <capacity>, Integer {1 to 2147483647}

PS_CLI/Applications/WebContainer/Tomcat/Executors/AJP/Queue> set capacity 0
*** Warning: Broadworks needs to be restarted for the changes to take effect
***

PS_CLI/Applications/WebContainer/Tomcat/Executors/AJP/Queue> get
  capacity = 0

PS_CLI/Applications/WebContainer/Tomcat/Executors/AJP/Queue>
```

```
PS_CLI/Applications/WebContainer/Tomcat/Executors/AJP/ThreadPool> help set
The command is used to modify ThreadPool settings.

Parameters description:
attribute    : The name of an attribute to modify.
min          : This parameter specifies the minimum number of threads to keep
               in steady state. Lower thread counts are possible during
               initialization.
max          : This parameter specifies the maximum number of threads this
pool
               can contain.
keepALiveTime: This parameter specifies the amount of idle time before an
               excess thread is terminated.

======================================================================
set
    <attribute>, Multiple Choice = {min, max, keepALiveTime}
```

```
        <min>, Integer {1 to 10000}
        <max>, Integer {1 to 10000}
        <keepALiveTime>, Integer {1 to 3600}

PS_CLI/Applications/WebContainer/Tomcat/Executors/AJP/ThreadPool> set min 50
max 500 keepAliveTime 1800
*** Warning: Broadworks needs to be restarted for the changes to take effect
***

PS_CLI/Applications/WebContainer/Tomcat/Executors/AJP/ThreadPool> get
  min = 50
  max = 500
  keepALiveTime = 1800

PS_CLI/Applications/WebContainer/Tomcat/Executors/AJP/ThreadPool>
```

8.4 Configuration of WebContainer

The WebContainer context allows configuration of Apache and Tomcat. For Apache, the configuration is contained within the General Settings context. For Tomcat, the configuration spans several distinct contexts, which are documented in the following sub-sections.

8.4.1 General Settings

The WebContainer general settings are configurable through the CLI, under the following levels:

- *PS_CLI/Applications/WebContainer/Apache/GeneralSettings*
- *PS_CLI/Applications/WebContainer/Tomcat/GeneralSettings*

8.4.1.1 Apache

The Apache server used by the Profile Server issues logs to the */var/broadworks/logs/apache* directory. The general settings for Apache allow configuring the log level.

The *GeneralSettings* level also allows configuring other apache parameters as follows:

- **maxQueuedConnection**: This parameter specifies the maximum length of the queue of pending connections.
- **usableWorkerThreads**: This parameter specifies the number of configured usable worker threads.
- **threadsPerWorker**: This parameter specifies the number of threads created by each child process.
- **statisticsRefreshPeriod**: This parameter specifies the frequency (in seconds) at which BroadWorks fetches PMs from the Apache Web Server.

Following is an example of setting the apache log level to “info”.

```
PS_CLI/Applications/WebContainer/Apache/GeneralSettings> help set
The command is used to modify Apache general settings.

Parameters description:
attribute              : The name of an attribute to modify.
logLevel               : This parameter specifies the log level.
maxQueuedConnection    : This parameter specifies the maximum length of the
                         queue of pending connections.
usableWorkerThreads    : This parameter specifies the number of configured
                         usable worker threads.
```

```
threadsPerWorker       : This parameter specifies the number of threads
created
                         by each child process.
statisticsRefreshPeriod: This parameter specifies the frequency at which
                         BroadWorks fetches PMs from the Apache Web Server.

=====================================================================
set
    <attribute>, Multiple Choice = {logLevel, maxQueuedConnection,
usableWorkerThreads, threadsPerWorker, statisticsRefreshPeriod}
        <logLevel>, Choice = {debug, info, notice, warn, error, crit, alert,
emerg}
        <maxQueuedConnection>, Integer {1 to 2147483647}
        <usableWorkerThreads>, Integer {100 to 10000}
        <threadsPerWorker>, Integer {25 to 10000}
        <statisticsRefreshPeriod>, Integer {1 to 86400}

PS_CLI/Applications/WebContainer/Apache/GeneralSettings> set logLevel info
*** Warning: Broadworks needs to be restarted for the changes to take effect
***

PS_CLI/Applications/WebContainer/Apache/GeneralSettings>

PS_CLI/Applications/WebContainer/Tomcat/OverloadProtection/Webapps> get
  logLevel = info
  maxQueuedConnection = 551
  usableWorkerThreads = 1000
  threadsPerWorker = 25
  statisticsRefreshPeriod = 5
```

To configure the worker busy threshold (which the general settings for Apache allow configuring), see section *8.5.4 Thresholds*.

8.4.1.2 Tomcat

The log settings for the Tomcat Application container are configurable through the CLI under the *PS_CLI/Applications/WebContainer/Tomcat/Logging* level. It uses the common BroadWorks logging mechanism for input and output channels.

The tomcat input channel includes the following names:

- Generic
- NameService
- TomcatCore
- CtiConnector
- GcLog
- SMAP
- ExternalAuthenticator

By default, the Tomcat application container used by the Profile Server issues logs in the */var/broadworks/logs/tomcat* directory. This log contains the container internal logs and possibly the exceptions it encounters while executing the various web applications.

The tomcat output channel follows the typical BroadWorks output channel pattern.

Following is an example of setting the TomcatCore log level to “FieldDebug”.

```
PS_CLI/Applications/WebContainer/Tomcat/Logging/InputChannels> h set
This command is used to modify InputChannels-related attributes in the
application.
```

```
Parameters description:
name     : Defines the type of input channel.
attribute: Identifies the queues to be modified.
enabled  : Enable/disable logging for a specific InputChannel.
severity : Define the minimum log level severity for a specific InputChannel.

=====================================================================
set
    <name>, Choice = {Generic, NameService, TomcatCore, CtiConnector, GcLog}
    <attribute>, Multiple Choice = {enabled, severity}
        <enabled>, Choice = {false, true}
        <severity>, Choice = {Debug, FieldDebug, Info, Notice, Warn}

PS_CLI/Applications/WebContainer/Tomcat/Logging/InputChannels> set
TomcatCore enabled true severity FieldDebug
...Done

PS_CLI/Applications/WebContainer/Tomcat/Logging/InputChannels> get
                  Name  Enabled    Severity
=============================================
               Generic     true
           NameService     true        Info
            TomcatCore     true  FieldDebug
          CtiConnector     true        Info
                 GcLog     true        Info
                  SMAP     false
 ExternalAuthenticator     true        Info

7 entries found.

PS_CLI/Applications/WebContainer/Tomcat/Logging/InputChannels>
```

8.4.2 Executors

Individual executors are available for each access point:

- AJP: Blocking HTTP requests coming through Apache.
- HTTPNIO: NonBlocking HTTP requests used for streaming.
- CTI: CTI interface.

The recommended operating model for the HTTPNIO and CTI executors is the fixed threading configuration described in section *8.3.2 Typical Configurations*: minPoolSize=maxPoolSize, with queueCapacity > 1.

This is configurable from the *PS_CLI/Applications/WebContainer/Tomcat/Executors* level.

8.4.2.1 Apache JServ Protocol (AJP) Executor

The AJP interface operates differently than the other two interfaces. For each concurrent HTTP request received on one of its client connections, Apache requires an AJP socket to Tomcat. These sockets are maintained as a pool and are re-used. In the WebContainer, each incoming AJP socket grabs a processing thread from the executor and remains bound to it until the socket is torn down. This executor thread will process all requests from that socket. The executor's thread pool can be sized to match the number of clients allowed on Apache or to a lower value (since not all clients issue concurrent requests). If there are more concurrent HTTP requests than there are threads available at the time, the additional requests are rejected with a 502 status code. This is an indication that the AJP's thread pool size should be increased.

The recommended operating model for the AJP executor is the Adjustable model as described in section *8.3.2 Typical Configurations*: minPoolSize < maxPoolSize, queueCapacity=0.

8.4.3 Process Metrics

The process metrics targets the Java Virtual Machine (JVM) metrics, which is used by many BroadWorks servers.

Java Virtual Machine (JVM) metrics provide several built-in statistics (using JVM MBeans) to report performance on key performance factors.

The Profile Server renders a CLI level for the process metrics under the *PS_CLI/Applications/WebContainer/Tomcat/JVMStatsCollector/GeneralSettings* level.

Data collection is controlled by setting the *enabled* attribute and data frequency is controlled by setting the period in seconds using the *statisticsRefreshPeriod* attribute.

- **enabled**: This parameter enables or disables data collection for JVM statistics.
- **statisticsRefreshPeriod**: The frequency (expressed in seconds) at which BroadWorks fetches PMs from the JVM MBeans. The default value is 5 seconds. In addition to the regular polling, the system refreshes the PMs after every full garbage collection[1].

8.4.3.1 CLI Context

Following is an example of setting the refresh period for the JVM Stats Collector.

```
PS_CLI/Applications/WebContainer/Tomcat/JVMStatsCollector/GeneralSettings>
help set
The command is used to modify JVM statistics collector general settings.

Parameters description:
attribute              : The name of an attribute to modify.
enabled                : This parameter enables or disables data collection
for
                         JVM statistics
statisticsRefreshPeriod: This parameter specifies the frequency at which
                         BroadWorks fetches PMs from JVM Mbeans

========================================================================
set
    <attribute>, Multiple Choice = {enabled, statisticsRefreshPeriod}
        <enabled>, Choice = {false, true}
        <statisticsRefreshPeriod>, Integer {1 to 86400}

PS_CLI/Applications/WebContainer/Tomcat/JVMStatsCollector/GeneralSettings>
PS_CLI/Applications/WebContainer/Tomcat/JVMStatsCollector/GeneralSettings>
set enabled true statisticsRefreshPeriod 10
*** Warning: Broadworks needs to be restarted for the changes to take effect
***

PS_CLI/Applications/WebContainer/Tomcat/JVMStatsCollector/GeneralSettings>
get
  enabled = true
  statisticsRefreshPeriod = 10

PS_CLI/Applications/WebContainer/Tomcat/JVMStatsCollector/GeneralSettings>
```

[1] There is a notification sent upon completion of a full garbage collection.

8.4.4 Overload Protection

The BWCommunicationUtility allows rate protection at the Web Container level. It is registered globally and allows overload controls before authentication takes place. This mechanism provides both global and webapp-level protection.

The Profile Server renders a CLI level for the overload protection under the *PS_CLI/Applications/WebContainer/Tomcat/OverloadProtection/Server* and *PS_CLI/Applications/WebContainer/Tomcat/OverloadProtection/Webapps* levels.

The Server level controls the number of requests allowed globally and the Webapps level controls the number of requests allowed per web app.

Each rate is controlled by first setting the calculation period in seconds using the p*eriod* attribute. Then the total number of requests allowed during this period is determined by the *limit* attribute.

For example:

- server.period = 10 seconds
- server.limit = 1000
- webapp.name = ECLQuery
- webapp.period = 10 seconds
- webapp.limit = 200
- global rate = 1000 requests every 10 seconds
- ECLQuery webapp rate = 200 requests every 10 seconds

8.4.4.1 CLI Context

Following is an example of limiting the number of request for the Xsi-Actions webapp.

```
PS_CLI/Applications/WebContainer/Tomcat/OverloadProtection/Webapps> help set
The command is used to modify per web application overload protection
settings.

Parameters description:
name     : This parameter specifies the name of the web application.
attribute: The name of an attribute to modify.
period   : This parameter specifies the duration expressed in seconds during
           which the total number of transactions is limited for the overall
           server.
limit    : This parameter specifies the maximum number of requests to be
           allowed during the specified period duration.

======================================================================
set
    <name>, Choice = { BroadworksFileRepos, BroadworksFileReposExtdCapture,
CCReporting, CCReportingRepository, DeviceManagementFiles, ECLQuery,
ECLReportingRepository, LogRepository, MediaFiles,
MeetMeConferencingRepository, MessageArchive}
    <attribute>, Multiple Choice = {period, limit}
        <period>, Integer {1 to 300}
        <limit>, Integer {1 to 65535}

PS_CLI/Applications/WebContainer/Tomcat/OverloadProtection/Webapps>
PS_CLI/Applications/WebContainer/Tomcat/OverloadProtection/Webapps> set
ECLQuery period 1 limit 50
*** Warning: Broadworks needs to be restarted for the changes to take effect
***

PS_CLI/Applications/WebContainer/Tomcat/OverloadProtection/Webapps> get
```

```
             Name  Period  Limit
================================
       ECLQuery      1     50

3 entries found.
```

8.4.5 Session Management

The session management allows specifying session timeouts globally and per webapp (optionally).

The Profile Server renders a CLI level for the session management under the *PS_CLI/Applications/WebContainer/Tomcat/ SessionManagement/Server* and *PS_CLI/ Applications/WebContainer/Tomcat/ SessionManagement/Webapps* levels.

Data collection is controlled by setting the *sessionTimeout* attribute:

- **sessionTimeout**: a duration expressed in seconds after which an inactive session times out. The default is 30.

The parameter is configurable per individual webapp (identified by the display name). When set, the value supersedes the value specified globally for the webapp. When there are multiple instances of a given webapp, they all share the same settings. That is, the same session timeout setting apply individually to all instances.

- **sessionTimeout (webapp)**: a webapp specific duration expressed in seconds after which an inactive session times out. By default, this parameter is empty for all configured webapps.

8.4.5.1 CLI Context

Following is an example of setting the Xsi-Actions timeout to “1800”.

```
PS_CLI/Applications/WebContainer/Tomcat/SessionManagement/Webapps> help set
The command is used to modify per web application session management
settings.

Parameters description:
name          : This parameter specifies the name of the web application.
attribute     : The name of an attribute to modify.
sessionTimeout: This parameter specifies a web application specific duration
                expressed in seconds after which an inactive session times
out.

=========================================================================
set
    <name>, Choice = { BroadworksFileRepos, BroadworksFileReposExtdCapture,
CCReporting, CCReportingRepository, DeviceManagementFiles, ECLQuery,
ECLReportingRepository, LogRepository, MediaFiles,
MeetMeConferencingRepository, MessageArchive}
    <attribute>, Multiple Choice = {sessionTimeout}
        <sessionTimeout>, Integer {1 to 86400}

PS_CLI/Applications/WebContainer/Tomcat/SessionManagement/Webapps> set Xsi-
Actions sessionTimeout 1200
*** Warning: Broadworks needs to be restarted for the changes to take effect
***

PS_CLI/Applications/WebContainer/Tomcat/SessionManagement/Webapps> get
                 Name  Session Timeout
=======================================
  BroadworksFileRepos             1200

1 entry found.
```

8.4.6 Thresholds

This section describes the list of configurable thresholds. For the list of SNMP traps generated upon a threshold crossing, see section *8.5.4 Thresholds*.

8.4.6.1 Apache

The Profile Server provides the ability to set a threshold for the http *workers busy ratio*. By default, this threshold defines the arm level to be 90 and the clear level to be 81. This threshold is enabled.

The *workers busy ratio* is defined as a percentage using the following formula: bwHttpWorkersBusy / bwHttpWorkersUsable * 100.

The Profile Server renders a CLI level for this threshold under the *PS_CLI/Applications/WebContainer/Apache/GeneralSettings/WorkersBusyThreshold* level.

The following is an example of setting the arm value to “88” and the reset value to “78”.

```
PS_CLI/Applications/WebContainer/Apache/GeneralSettings/WorkersBusyThreshold> help
set
The command is used to modify threshold settings.

Parameters description:
attribute : The name of an attribute to modify.
enabled   : Specify if a threshold is active or not.
armValue  : Value for enabling a threshold crossing.
resetValue: Value for clearing a threshold crossing.

======================================================================
set
    <attribute>, Multiple Choice = {enabled, armValue, resetValue}
        <enabled>, Choice = {false, true}
        <armValue>, Integer {0 to 100}
        <resetValue>, Integer {0 to 100}

PS_CLI/Applications/WebContainer/Apache/GeneralSettings/WorkersBusyThreshold>
PS_CLI/Applications/WebContainer/Apache/GeneralSettings/WorkersBusyThreshold> set
armValue 88 resetValue 78
*** Warning: Broadworks needs to be restarted for the changes to take effect ***

PS_CLI/Applications/WebContainer/Apache/GeneralSettings/WorkersBusyThreshold> get
  enabled = true
  armValue = 88
  resetValue = 78

PS_CLI/Applications/WebContainer/Apache/GeneralSettings/WorkersBusyThreshold>
```

8.4.6.2 Executors

For each executor name (AJP, CTI, and HttpNio), the Profile Server provides the ability to set the following thresholds.

Executor Queue Usage Ratio

The executor queue usage ratio is defined as a percentage using the following formula: EventQueueCount / queueCapacity * 100. By default, this threshold defines the arm level to be 90 and the clear level to be 81. This threshold is enabled.

The Profile Server renders a CLI level for this threshold under the *PS_CLI/Applications/WebContainer/Tomcat/Executors/<executorName>/Queue/SizeThreshold* level.

The following is an example of setting the arm value to “88” and the reset value to “78” for the AJP executor.

```
PS_CLI/Applications/WebContainer/Tomcat/Executors/AJP/Queue/SizeThreshold> help set
The command is used to modify threshold settings.

Parameters description:
attribute : The name of an attribute to modify.
enabled   : Specify if a threshold is active or not.
armValue  : Value for enabling a threshold crossing.
resetValue: Value for clearing a threshold crossing.

======================================================================
set
    <attribute>, Multiple Choice = {enabled, armValue, resetValue}
        <enabled>, Choice = {false, true}
        <armValue>, Integer {0 to 100}
        <resetValue>, Integer {0 to 100}

PS_CLI/Applications/WebContainer/Tomcat/Executors/AJP/Queue/SizeThreshold> set
armValue 88 resetValue 78
*** Warning: Broadworks needs to be restarted for the changes to take effect ***

PS_CLI/Applications/WebContainer/Tomcat/Executors/AJP/Queue/SizeThreshold> get
  enabled = true
  armValue = 88
  resetValue = 78

PS_CLI/Applications/WebContainer/Tomcat/Executors/AJP/Queue/SizeThreshold>
```

Executor Queue Latency

The executor queue latency is defined as an absolute value and represents the number of requests that are currently queued while waiting for other tasks to complete during execution. By default, this threshold defines the arm level to be 2500 and the clear level to be 2000. This threshold is enabled.

The Profile Server renders a CLI level for this threshold under the *PS_CLI/Applications/WebContainer/Tomcat/Executors/<executorName>/Queue/LatencyThreshold/* level.

Executor Thread Pool Processing Time

The executor thread pool processing time is defined as an absolute value and represents the execution time taken by a task to complete. By default, this threshold defines the arm level to be 2500 and the clear level to be 2000. This threshold is enabled.

The Profile Server renders a CLI level for this threshold under the *PS_CLI/Applications/WebContainer/Tomcat/Executors/<executorName>/ThreadPool/ProcessingTimeThreshold/* level.

Executor Thread Pool Busy Ratio

The executor thread pool busy ratio is defined as a percentage using the following formula: threadsBusy / maxPoolSize * 100. By default, this threshold defines the arm level to be 95 and the clear level to be 85. This threshold is enabled.

The Profile Server renders a CLI level for this threshold under the *PS_CLI/Applications/WebContainer/Tomcat/Executors/<executorName>/ThreadPool/UsageThreshold/* level.

8.4.6.3 Process Metrics

Heap and Nonheap Memory Usage Ratios

The Profile Server provides the ability to set a threshold for the heap and nonheap memory usage ratios. By default, these thresholds define the arm level to be 90 and the clear level to be 81. These thresholds are enabled.

The heap memory usage ratio is defined as a percentage and represents the heap memory usage after a full garbage collection.

The nonheap memory usage ratio is defined as a percentage and represents the nonheap memory usage after a full garbage collection.

The Profile Server renders a CLI level for this threshold under the *PS_CLI/Applications/WebContainer/Tomcat/JVMStatsCollector/HeapUsageThreshold/* level.

The Profile Server renders a CLI level for this threshold under the *PS_CLI/Applications/WebContainer/Tomcat/JVMStatsCollector/NonHeapUsageThreshold* / level.

The following is an example of setting the arm value to “88” and the reset value to “78” for the heap memory usage threshold.

```
PS_CLI/Applications/WebContainer/Tomcat/JVMStatsCollector/HeapUsageThreshold> help
set
The command is used to modify threshold settings.

Parameters description:
attribute : The name of an attribute to modify.
enabled   : Specify if a threshold is active or not.
armValue  : Value for enabling a threshold crossing.
resetValue: Value for clearing a threshold crossing.

======================================================================
set
    <attribute>, Multiple Choice = {enabled, armValue, resetValue}
        <enabled>, Choice = {false, true}
        <armValue>, Integer {0 to 100}
        <resetValue>, Integer {0 to 100}

PS_CLI/Applications/WebContainer/Tomcat/JVMStatsCollector/HeapUsageThreshold> set
armValue 88 resetValue 78
*** Warning: Broadworks needs to be restarted for the changes to take effect ***

PS_CLI/Applications/WebContainer/Tomcat/JVMStatsCollector/HeapUsageThreshold> get
  enabled = true
  armValue = 88
  resetValue = 78

PS_CLI/Applications/WebContainer/Tomcat/JVMStatsCollector/HeapUsageThreshold>
```

8.5 Management of WebContainer

The combination of flexible configurations and strong reporting capabilities allows configuring specific thresholds with pre-defined values. Upon a threshold crossing, the system generates SNMP traps to report this state to the network management layers.

The Profile Server allows enabling or disabling of data collection and the frequency of these metrics as well as customizing thresholds.

From a network management layer, it is possible to collect the Performance Measurements (PMs) and receive threshold crossing notifications.

8.5.1 Metric Groups

For a review of the metric groups, see section *7.2.1.2 WebContainer Metric Groups*.

8.5.2 Sequence Diagrams

Using sequence diagrams, this section highlights key processing time metrics. They are organized by connector and executor types.

The key difference among the connector types is the parallel processing. That is, for the AJP executor, the processing blocks until a response is received whereas for the HttpNio and CTI executors, the processing returns immediately after an event is sent. The following sequence diagrams also capture the implications of the Open Client Interface (OCI) executor, which is involved when receiving responses from other BroadWorks servers.

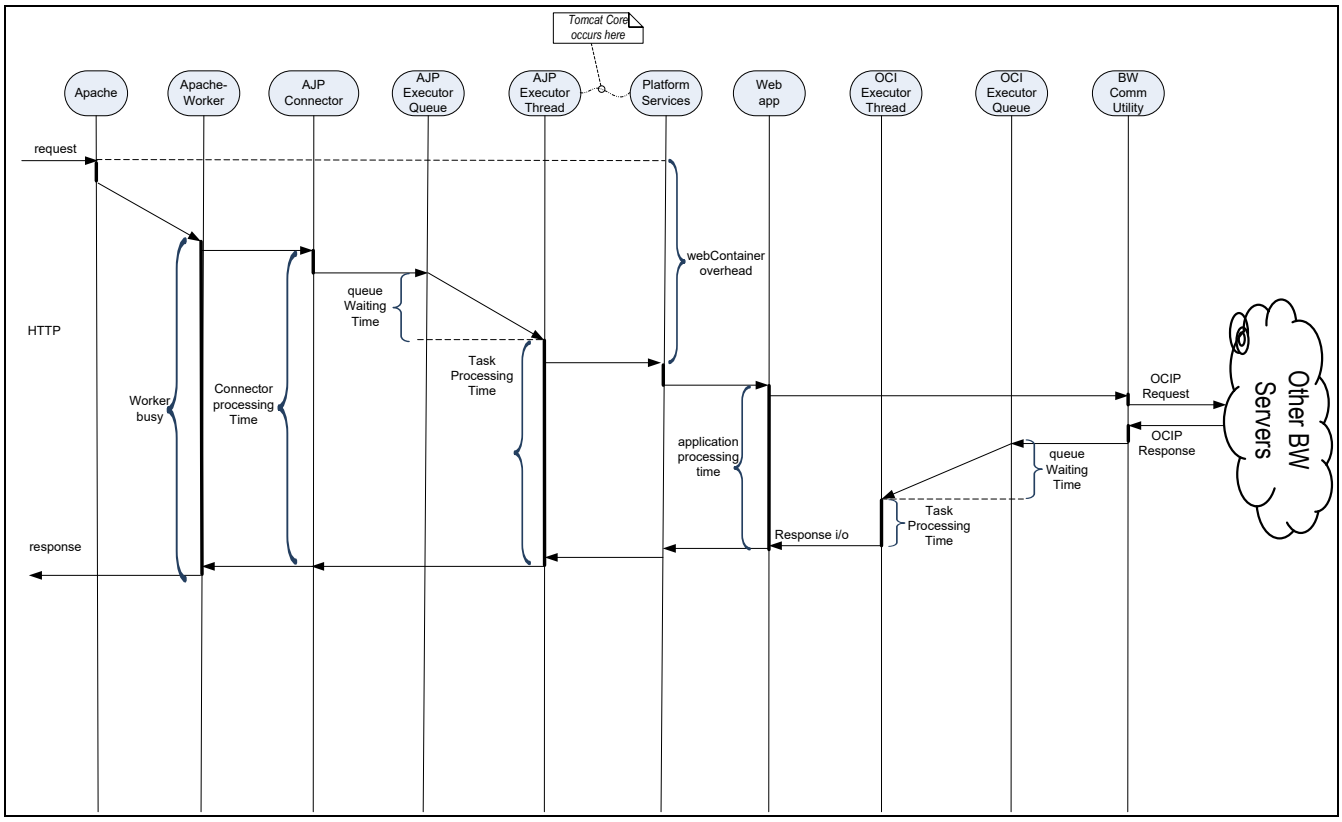

Figure 3 AJP Sequence Diagram

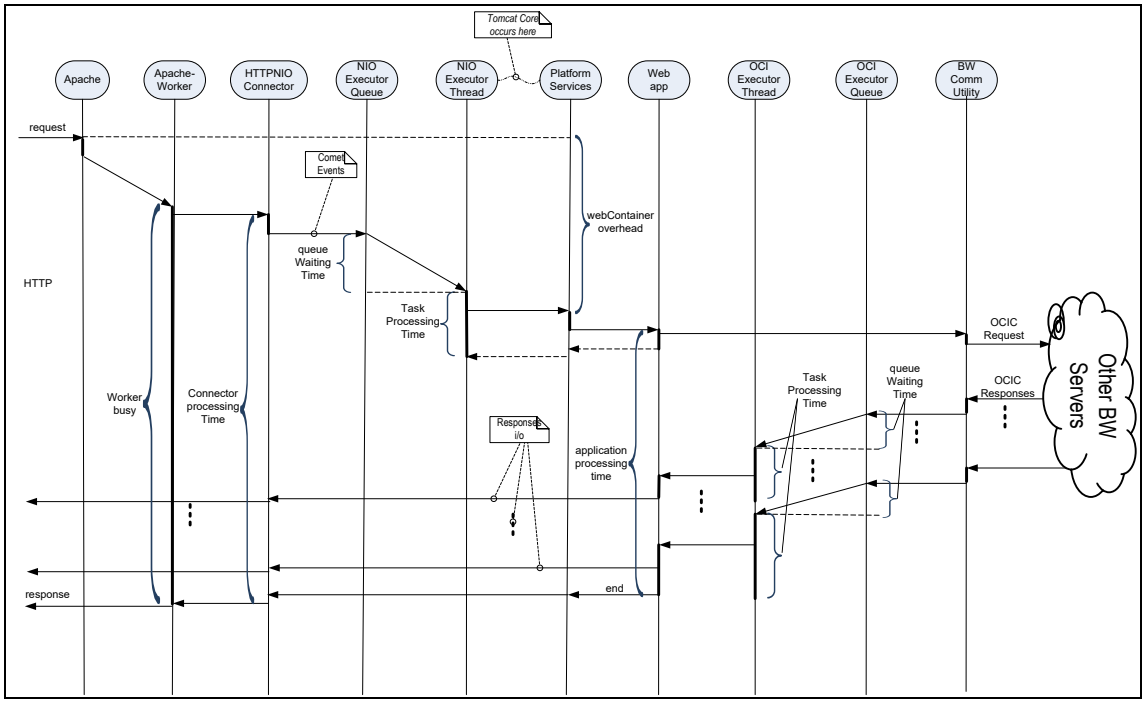

Figure 4 HttpNio Sequence Diagram

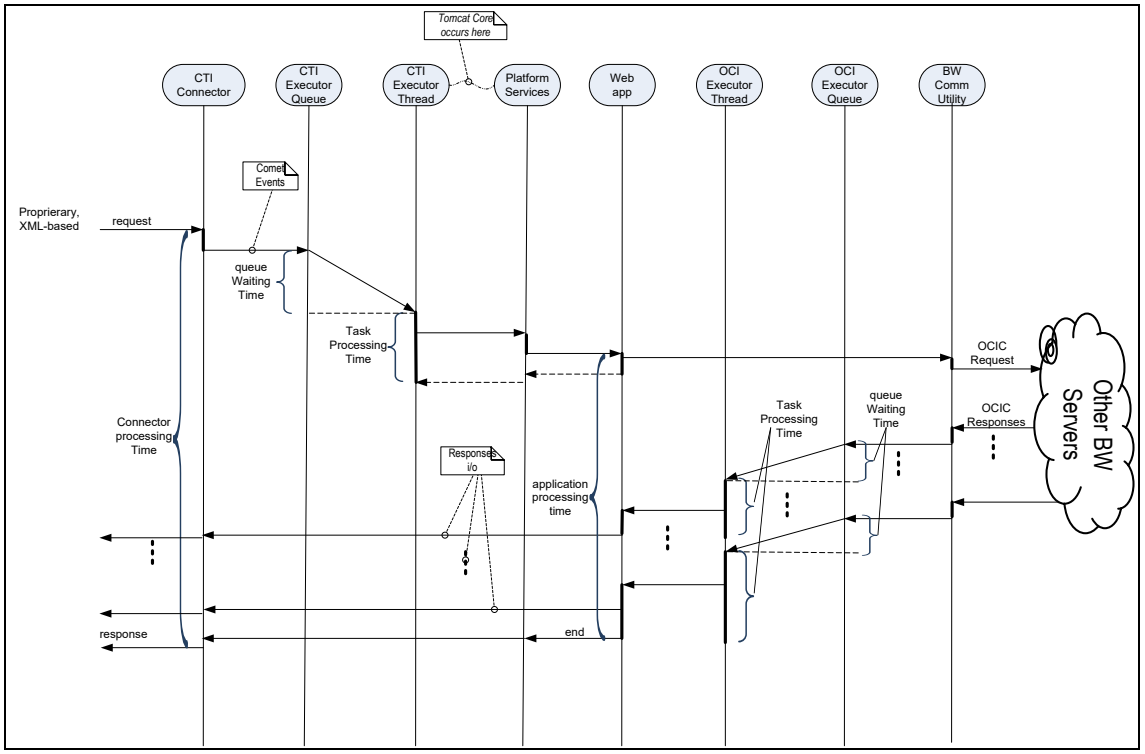

Figure 5 CTI Sequence Diagram

These diagrams highlight the metrics described in the following subsections. Note that only the objects identifiers (OIDs) pertinent to the key metrics are presented here. For full details, see the *BroadWorks Performance Measurements Interface Specification* [9].

8.5.2.1 Worker Busy

This statistic reports the time spent in the apache worker. The *BW-WebContainer MIB* reports this statistic with the following OID:

- bwWebContainer.protocols.http.serverResources.workers.bwHttpWorkersBusy

This OID is available from the following CLI level.

```
PS_CLI/Monitoring/PM/WebContainer> cd protocols/http/serverResources/workers
bwWebContainer/protocols/http/serverResources/workers/

PS_CLI/Monitoring/PM/WebContainer> get
------------------------------------------------------------------------------
bwWebContainer/protocols/http/serverResources/workers/
------------------------------------------------------------------------------
                          bwHttpWorkerThreadsBusy           1
                          bwHttpWorkerThreadsIdle          49
     bwHttpWorkerThreadsConsecutiveFailedRefresh           0
                        bwHttpWorkerThreadsUsable        1000
                       bwHttpWorkerThreadsBusyAvg           1
                       bwHttpWorkerThreadsBusyMax           7

PS_CLI/Monitoring/PM/WebContainer>
```

8.5.2.2 WebContainer Overhead

This statistic reports the WebContainer core traversal time on incoming requests as defined by the time differential between a request's arrival at the Apache Server and its arrival at a webapp. The system does not report an instant value. Instead, it tracks the minimum, average, and maximum values. The *BW-WebContainer MIB* reports this statistic with the following OIDs:

- bwWebContainer.protocols.http.serverResources.overhead.bwHttpOverheadTimeMin
- bwWebContainer.protocols.http.serverResources.overhead.bwHttpOverheadTimeAvg
- bwWebContainer.protocols.http.serverResources.overhead.bwHttpOverheadTimeMax

These OIDs are available from the following CLI level.

```
PS_CLI/Monitoring/PM/WebContainer> cd protocols/http/serverResources/overhead
bwWebContainer/protocols/http/serverResources/overhead/

PS_CLI/Monitoring/PM/WebContainer> get
------------------------------------------------------------------------------
bwWebContainer/protocols/http/serverResources/overhead/
------------------------------------------------------------------------------
              bwHttpOverheadTimeSamples          27
                  bwHttpOverheadTimeAvg          89
                  bwHttpOverheadTimeMin           1
                  bwHttpOverheadTimeMax        1051
     bwHttpOverheadTimeMaxTimestampMSB      314477
     bwHttpOverheadTimeMaxTimestampLSB  2172958808

PS_CLI/Monitoring/PM/WebContainer>
```

8.5.2.3 Connector Processing Time

This statistic measures the processing time spent in the connector. The system does not report an instant value. Instead, it reports the cumulative time and the maximum value for an individual request. The *BW-WebContainer MIB* reports this statistic with the following OIDs:

- bwWebContainer.connectors.bwConnectorTotalProcessingTime
- bwWebContainer.connectors.bwConnectorMaxProcessingTime

These OIDs are available from the following CLI level.

```
PS_CLI/Monitoring/PM/WebContainer> cd connectors;get
bwWebContainer/connectors/
------------------------------------------------------------------------------
bwWebContainer/connectors/
------------------------------------------------------------------------------
bwConnectorTable:

  (1) bwConnectorIndex
  (2) bwConnectorName
  (3) bwConnectorExecutorName
  (4) bwConnectorRequestCount
  (5) bwConnectorBytesRx
  (6) bwConnectorBytesTx
  (7) bwConnectorTotalProcessingTime
  (8) bwConnectorMaxProcessingTime

   (1)                          (2)       (3)     (4)     (5)      (6)     (7)     (8)
     1  [localhost/127.0.0.1]:8010  httpnio       0       0        0       0       0
     2  [localhost/127.0.0.1]:8009      ajp      36     120   225548   10118    4191

PS_CLI/Monitoring/PM/WebContainer>
```

8.5.2.4 Executors

The Queue Waiting Time and Task Processing Time are part of the executor metrics.

The Queue Waiting Time statistic reports the time spent waiting in the executor queue. A delay occurs when the executor threads are all busy processing other events. This metric is reported in a MIB table identified by the executor type. The *BW-WebContainer MIB* reports this statistic with the following OID:

- bwWebContainer.executors.executorTable.bwExecutorQueueWaitingTime

The Task Processing time statistic reports the processing time spent in the executor thread. This metric is reported in a MIB table identified by the executor type. The system does not report an instant value. Instead, it tracks the minimum, average, and maximum values. The *BW-WebContainer MIB* reports this statistic with the following OIDs:

- bwWebContainer.executors.executorTable.bwExecutorTaskProcessingTimeMin
- bwWebContainer.executors.executorTable.bwExecutorTaskProcessingTimeAvg
- bwWebContainer.executors.executorTable.bwExecutorTaskProcessingTimeMax

These OIDs are available from the following CLI level.

```
PS_CLI/Monitoring/PM/WebContainer> cd executors;get
bwWebContainer/executors/
------------------------------------------------------------------------------
bwWebContainer/executors/
------------------------------------------------------------------------------
    *bwExecutorReset           0
bwExecutorTable:
```

```
  (1) bwExecutorIndex
  (2) bwExecutorName
  (3) bwExecutorQueueCapacity
  (4) bwExecutorQueueSize
  (5) bwExecutorQueueWaitingTimeAvg
  (6) bwExecutorQueueWaitingTimeMin
  (7) bwExecutorQueueWaitingTimeMax
  (8) bwExecutorQueueWaitingTimeMaxTimestampMSB
  (9) bwExecutorQueueWaitingTimeMaxTimestampLSB
 (10) bwExecutorMinPoolSize
 (11) bwExecutorMaxPoolSize
 (12) bwExecutorThreadsBusy
 (13) bwExecutorCompletedTaskCount
 (14) bwExecutorThreadsAlive
 (15) bwExecutorThreadsAliveMax
 (16) bwExecutorTaskProcessingTimeAvg
 (17) bwExecutorTaskProcessingTimeMin
 (18) bwExecutorTaskProcessingTimeMax
 (19) bwExecutorTaskProcessingTimeMaxTimestampMSB
 (20) bwExecutorTaskProcessingTimeMaxTimestampLSB

(1)     (2)    (3) (4) (5) (6) (7) (8)        (9) (10) (11) (12) (13) (14) (15) (16) (17) (18) (19) (20)
  1     ajp 20000   0 248   0 950 314 2050872259  200  200    5    0    5    5    0    0    0    0    0
  2 httpnio 20000   0   0   0   0   0          0  200  200    0    0    0    0    0    0    0    0    0

PS_CLI/Monitoring/PM/WebContainer>
```

8.5.2.5 Application Processing Time

This statistic reports the processing time spent in the web application. The system does not report an instant value. Instead, it reports only the cumulative time. The *BW-WebContainer MIB* reports this statistic with the following OID:

- bwWebContainer.applications.applicationResources.bwApplicationResourceTotalProcessingTime

This OID is available from the following CLI level.

```
PS_CLI/Monitoring/PM/WebContainer> cd applications/applicationResources;get
bwWebContainer/applications/applicationResources/
------------------------------------------------------------------------------
bwWebContainer/applications/applicationResources/
------------------------------------------------------------------------------
bwApplicationResourcesTable:

  (1) bwApplicationResourceIndex
  (2) bwApplicationResourceWebAppName
  (3) bwApplicationResourceRequestCount
  (4) bwApplicationResourceTotalProcessingTime

    (1)                         (2)     (3)     (4)

    (1)                     (2)     (3)     (4)
      1  /MeetMeConfRepository     611     471
      2                      /    6866   42455
      3            /MediaFiles       0       0
      4               /dmfiles       0       0
      5            /ExtCapture      60      85

PS_CLI/Monitoring/PM/WebContainer> >
```

8.5.3 Process Metrics

The process metrics targets the Java Virtual Machine (JVM) metrics, which is used by many BroadWorks servers.

The JVM metrics provide several built-in statistics (using JVM MBeans) to report performance on key performance factors.

The *BW-WebContainer MIB* reports this statistic with the following OID:

- bwWebContainer.processMetrics

8.5.3.1 Memory

The OIDs for heap and nonheap memories are available from the following CLI levels.

```
PS_CLI/Monitoring/PM/WebContainer> cd processMetrics/memory/heap;get
bwWebContainer/processMetrics/memory/heap/
------------------------------------------------------------------------------
bwWebContainer/processMetrics/memory/heap/
------------------------------------------------------------------------------
                  bwProcessMetricsHeapInitSize   226492416
                   bwProcessMetricsHeapMaxSize   226492416
                      bwProcessMetricsHeapUsed    60406032
                   bwProcessMetricsHeapUsedMax    92134800
                 bwProcessMetricsHeapCommitted   226492416
     bwProcessMetricsHeapLastPostCollectionSize    60496664

PS_CLI/Monitoring/PM/WebContainer>
```

```
PS_CLI/Monitoring/PM/WebContainer> cd memory/nonheap;get
bwWebContainer/processMetrics/memory/nonheap/
------------------------------------------------------------------------------
bwWebContainer/processMetrics/memory/nonheap/
------------------------------------------------------------------------------
                  bwProcessMetricsNonHeapInitSize    24313856
                   bwProcessMetricsNonHeapMaxSize   318767104
                      bwProcessMetricsNonHeapUsed    34920144
                   bwProcessMetricsNonHeapUsedMax    38417936
                 bwProcessMetricsNonHeapCommitted    35192832
     bwProcessMetricsNonHeapLastPostCollectionSize           0

PS_CLI/Monitoring/PM/WebContainer>
```

8.5.3.2 Thread

The OIDs for threads are available from the following CLI level.

```
PS_CLI/Monitoring/PM/WebContainer> cd processMetrics/threads;get
bwWebContainer/processMetrics/threads/
------------------------------------------------------------------------------
bwWebContainer/processMetrics/threads/
------------------------------------------------------------------------------
        bwProcessMetricsThreadsAlive          33
     bwProcessMetricsThreadsAliveMax          34
      bwProcessMetricsThreadsStarted       37911


PS_CLI/Monitoring/PM/WebContainer>
```

8.5.4 Thresholds

This section describes the list of SNMP traps generated upon a threshold crossing. For the list of thresholds whose default values are configurable, see section *8.4.6 Thresholds*.

8.5.4.1 Apache

The Profile Server provides the ability to set a threshold for the *http workers busy* ratio. Upon a threshold crossing, the system generates the following trap:

- Trap *bwHttpWorkersBusyExceeded* (of severity high): the ratio of http workerBusy exceeded the threshold level

8.5.4.2 Executors

For each of the executor name (AJP, CTI, and httpnio), the Profile Server provides the ability to generate the following traps:

Executor Queue Usage Ratio

The executor queue usage ratio is defined as a percentage using the following formula: EventQueueCount / queueCapacity * 100. Upon a threshold crossing, the system generates the following trap:

- Trap *bwExecutorQueueUsageExceeded* (of severity high): the usage ratio of the executor queue exceeded the threshold level.

Executor Queue Latency

The executor queue latency is defined as an absolute value and represents the number of requests that are currently queued while waiting for other tasks to complete during execution. Upon a threshold crossing, the system generates the following trap:

- Trap *bwExecutorQueueLatencyExceeded* (of severity high): the executor queue latency exceeded the threshold level.

Executor Thread Pool Processing Time

The executor thread pool processing time is defined as an absolute value and represents the execution time taken by a task to complete. Upon a threshold crossing, the system generates the following trap:

- Trap *bwExecutorThreadPoolProcessingTimeExceeded* (of severity high): the executor thread pool processing time exceeded the threshold level.

Executor Thread Pool Busy Ratio

The executor thread pool busy ratio is defined as a percentage using the following formula: threadsBusy / maxPoolSize * 100. Upon a threshold crossing, the system generates the following trap:

- Trap *bwExecutorThreadPoolBusyExceeded* (of severity high): the executor thread pool busy ratio exceeded the threshold level.

8.5.4.3 Process Metrics

Heap and Nonheap Memory Usage Ratios

The Profile Server provides the ability to set a threshold for the heap and nonheap memory usage ratios.

The heap memory usage ratio is defined as a percentage and represents the heap memory usage after a full garbage collection.

The nonheap memory usage ratio is defined as a percentage and represents the nonheap memory usage after a full garbage collection.

Upon a threshold crossing, the system generates the following traps:

- Trap *bwHeapMemoryUsageExceeded* (of severity high): the memory usage ratio after a full garbage collection (in %) exceeded the threshold level.
- Trap *bwNonheapMemoryUsageExceeded* (of severity high): the nonheap memory usage ratio after a full garbage collection (in %) exceeded the threshold level.

8.5.4.4 Overload Protection

The Profile Server raises an SNMP trap when the system starts rejecting requests. For these traps, the threshold setting is embedded in the definition of the protection mechanism. Hence, there is no additional threshold parameter required. Since these traps are notifications, they do not clear automatically.

- Trap *bwServerTransactionLimitExceeded*: during the last overload protection period, the number of incoming requests exceeded the prescribed limit for the whole server.
- Trap *bwWebAppTransactionLimitExceeded*: during the last overload protection period, the number of incoming requests exceeded the prescribed limit for a specific webapp (as identified by its name).
- Trap *bwUserTransactionLimitExceeded*: during the last overload protection period, the number of incoming requests exceeded the prescribed limit for a specific user.

8.6 Application Management

Operators can activate and deploy applications on the Profile Server. The Profile Server provides application management functionality through the CLI, under the *Maintenance/ManagedObjects* level. Applications can be activated, deactivated, deployed, and undeployed. All operations are CLI-driven. No manual operations are required.

Applications and web applications are packaged using the BroadWorks Application format (*.bwar)*. All applications on the Profile Server are created and released by BroadSoft.

The full life cycle of an application is:

Activation → Deployment → Undeployment → Deactivation

Individual CLI operations are described in the following subsections.

Unless otherwise mentioned, no restart is necessary for any operation. All changes are effective immediately.

For detailed information and procedures on application management, see the *BroadWorks Xtended Services Platform Configuration Guide* [3].

8.7 HTTP Server Configuration

Typically, a Web Server processes HTTP requests on port 80 and Hypertext Transfer Protocol Secure Sockets (HTTPS) requests on port 443 of its public Internet Protocol (IP) interface. The Profile Server requires the additional capacity to serve HTTP/HTTPS on different ports and to support multiple IP interfaces. For this reason, the Profile Server allows complete operator control over which IP interfaces and ports are used to accept server requests. The CLI provides comprehensive management functions to configure and set up HTTP servers. All commands are located under the *PS_CLI/Interfaces/Http/HttpServer* level.

The Profile Server defines an HTTP server as a point of service for HTTP or HTTPS requests. An HTTP server is uniquely defined by an IP address of a local interface and a port (for example, 192.168.12.12:80). Each HTTP server is defined with a public host name and can be secured with a secure sockets layer (SSL).

During the Profile Server installation, a default secured HTTP server is created using the first IP address and port 443.

Requests received by all Web Servers are dispatched to the proper web application according to the web application's context path. There are no restrictions applied to the dispatching of requests based on the HTTP server that received them.

8.7.1 Public Host Name

Each HTTP server is configured with a public host name. The HTTP servers issue a redirect to *http://<hostname>:port/...* for each request they receive in which the host name part of the URL does not match its configured host name.

For example, a server configured with a host name of "myhost1.com" would redirect a request from *http://<ip>/some* to *http://myhost1.com/some*.

Clients must be able to resolve the host name to an IP address. The Profile Server verifies that the host name is locally defined in */etc/host*.

Exceptions to this behavior can be established using aliases. For more information, see section *8.7.6 HTTP Aliases*.

8.7.2 Secure HTTP Server

The HTTP servers can be configured as secured or not secured. A secured HTTP server only serves HTTPS requests.

Mutual TLS/SSL authentication or certificate-based mutual authentication refers to two parties authenticating each other through verifying the provided digital certificate so that both parties are assured of the other's identity. In BroadWorks terms, it refers to a client authenticating themselves to a Profile Server and the Profile Server also authenticating itself to the client through verifying each other's certificate. The Profile Server supports one-way authentication, authenticating itself to the client similar to any other web server and, optionally, supports mutual TLS/SSL authentication.

One-way or two-way authentication requires secure sockets layer (SSL) certificates signed by a recognized authority. On the server side, the Profile Server initially generates self-signed certificates and provides CLI commands to manage replacement of these self-signed certificates with properly signed ones. The use of self-signed certificates allows for immediate use of the HTTP server for HTTPS; however, they should be replaced with signed certificates for production systems. On the client side, there are commands to generate trust anchors.

8.7.3 Create HTTP Server

Additional HTTP servers can be added to the Profile Server by using the add command. The IP must be an IP that is locally valid. The host name is the name that clients are redirected to and shows up in URLs.

For example:

PS_CLI/Interface/Http/HttpServer> add 192.168.12.12 8080 myhost false

8.7.4 Delete HTTP Server

HTTP servers that are no longer required can be deleted using the delete command. The IP and port of the server to delete must be provided.

For example:

PS_CLI/Interface/Http/HttpServer> delete 192.168.12.12 8080

8.7.5 Manage SSL Certificates

8.7.5.1 Server Certificates

When using a secured HTTP server, the Profile Server automatically generates self-signed certificates using the host specified for that server. To sign the certificate, the certificate signing request must be extracted from the Profile Server. Using the "sslGenKey", the certificate signing request file is generated to the */tmp* directory under the name <host>.csr. This file is required for the official signed procedures.

Once signed, the certificate file must be locally copied to the Profile Server, for example, in */tmp*. The sslUpdate command can be used to reload the signed certificate (and optionally, the certificate chain file), replacing the old self-signed certificates. If a certificate chain file is no longer required, it can be removed using the sslRemove command.

The sslExport command can also be used to export the certificate file, the SSL key, and the certificate chain file.

8.7.5.2 Trust Anchors

Configuration of client authentication is centralized under a dedicated CLI level named *ClientAuthentication*. The list of trust anchors applies to all interfaces listed in the *HttpServer* level where the field *clientAuthReq* is set to "true". The following shows where this *HttpServer* level is located:

PS_CLI/Interface/Http/ClientAuthentication>

This level includes a parameter named *chainDepth* to control the maximum length of the chain of certificates. Note that this parameter is specific to Http interfaces and does not have any equivalency in the context of the CTI and OCS interfaces.

Requiring client authentication means that the server must validate the certificate received from the client during the SSL handshake. The server uses a trust store that contains trust anchors to validate these certificates. Therefore, this CLI level provides a sublevel named *trusts* that defines a set of dedicated commands for handling trust anchors. This sublevel is located as follows for the *HttpServer* level.

PS_CLI/Interface/Http/ClientAuthentication/Trusts>

These interactions allow the generation of a trust anchor that can be directly distributed to the client software[2] or can be used as an issuer for generating client certificates. Depending on the network requirement, a BroadSoft customer may require multiple trust anchors. For example, certain device vendors demand the use of their own certificates. The CLI commands require an administrator to specify an alias name to allow for identifying the trust anchors individually.

[2] A trust anchor is itself a certificate.

8.7.6 HTTP Aliases

The Profile Server allows the use of HTTP aliases for each local interface. Aliases are managed through the *PS_CLI/Interface/Http/HttpAliases* level through the CLI.

Normally, the HTTP servers receiving requests with a host name different from their public host name redirect the client to use the public host name. For more information, see section *8.7.1 Public Host Name*.

Aliases define additional host name values that the HTTP servers accept without redirections. Aliases can be added and deleted through the CLI.

8.7.6.1 Fully Qualified Domain Name Mapping

The HTTP servers support mapping of known fully qualified domain names (FQDNs) to aliases through redirections. An FQDN value can be provisioned on each HTTP alias. HTTP servers receiving a request with the FQDN as a host name redirect the client to use the corresponding alias.

8.7.7 HTTP Bindings

The HTTP bindings define which interfaces (IP address and port) a given web application is available and allow the binding of all applications to specific interfaces. The name of the web application corresponds to the display name defined in its *web.xml* file. An HTTP server web application binding can be created even if the application is not activated or deployed. By default, if no entry is defined under the HTTP server binding, the web application is available on all interfaces. As soon as one entry is added, it is only allowed on that interface.

8.8 Redundancy

The Profile Server is deployed in a farm architecture and each Profile Server uses *rsync* to exchange files between other Profile Servers in the farm.

To add a new peer, perform the following steps:

> **NOTE**: The new peer is identified as the *target* and the peer from which the *target* recovers is called the *source.*

1) Run the config-redundancy script on the new peer.
2) On the target, make sure that all peers are present on the replication peer list found under the CLI level *System/Peering/Peers*.
3) Make sure that BroadWorks is not running on the target.
4) Through the CLI of the target, go to *Application/<Application Name>/FileReplication/Tools* and execute the import command.
5) On every peer, add the target on the peer list found under the CLI level *System/Peering/Peers*.
6) Through the CLI of the target, go to *Application/<Application_Name>/FileReplication/Tools* and execute the syncPeer command.

Steps *4)* and *6)* need to be done for the following applications:

- BroadworksFileRepos
- BroadworksFileReposExtdCapture

- ECLReportingRepository
- MessageArchive

8.9 Configuration of BWCommunicationUtility

The BWCommunicationUtility provides the framework on which web applications can be built. It can be configured with default settings that are inherited by all web applications. However, web applications are not forced to use the default settings. Instead, they can override them with values that are more appropriate to their needs. The BWCommunicationUtility default settings are configurable through the CLI, under the *PS_CLI/System/CommunicationUtility/DefaultSettings* level.

8.9.1 Integration Mode

The integration mode used to communicate with other BroadWorks servers is controlled by the *mode* attribute. The mode can be set to either "NS" or "AS", where NS should be used when a Network Server is used to front end one or multiple Application Server clusters, and AS should be used when connecting to a single Application Server cluster without a Network Server.

- In *NS mode*, the Network Server cluster address must be provided as well as the Open Client Interface-Provisioning (OCI-P)/OCI-Call Control (OCI-C) ports to use. Note that in *NS mode*, all Application Server clusters must be using the same OCI-P port.
- In *AS mode*, the primary and secondary Application Server addresses are configurable, as well as the Open Client Interface ports to use for OCI-P.

8.9.2 Provision to Secondary

The Profile Server can be configured to give preference to the secondary Application Server when sending OCI-P messages to an Application Server cluster. The *provisionToSecondary* attribute controls this behavior.

8.9.3 Communication Settings

The following settings affect the communication channels to other BroadWorks servers:

- *reconnectionTimerSecs* – The number of seconds to wait between reconnection attempts to a BroadWorks server
- *responseTimeoutSecs* – The number of seconds to wait for a response from a BroadWorks Server after issuing a message
- *useSecureBCCT* – Only secure BCCT connections are used for OCI-P and OCI-C.

8.9.4 Overflow Protection Settings

The BWCommunicationUtility allows rate protection according to the web application for user HTTP/HTTPS requests.

The rate is controlled by first setting the calculation period in seconds using the *transactionLimitPeriodSecs* attribute. Then, the number of requests allowed per user during the same period is determined by the *userTransactionLimit*.

For example:

- transactionLimitPeriodSecs = 10 seconds
- userTransactionLimit = 5
- user rate = 5 requests every 10 seconds

The rates are computed according to the web application.

Note that section *8.4.4 Overload Protection* describes another mechanism that provides overload controls before authentication takes place.

8.9.5 Executors

8.9.5.1 General Guidelines

An executor is a means of bounding and managing resources consumed when executing a collection of tasks. The system consumes threads when executing a collection of tasks. As such, the executors are viewed as thread pools and categorized as OCI-C, OCI-P, AJP (part of BW-WebContainer), Http-Nio (part of BW-WebContainer) and CTI (part of BW-WebContainer). There is currently only one instance per executor type but it may change over time. For example, the Profile Server may eventually have OCI-P1 and OCI-P2.

To address the level of flexibility expected from the Profile Server, the following parameters are required:

- **minPoolSize**: defined as the expected number of concurrent threads running when a system in is in a stable state.
- **maxPoolSize**: defined as the maximum number of concurrent threads running when a system is under heavier processing load.
- **keepAliveTime**: defined as the amount of idle time before an excess thread is terminated.
- **queueCapacity**: defined as the maximum number of requests that can be queued while waiting for other tasks to complete their execution.

Based on the above parameters, the following approach is used when handling incoming web container requests:

- The actual pool size is automatically adjusted based on the **minPoolSize** and the **maxPoolSize**. When a new task is submitted and fewer than **minPoolSize** threads are running, the system allocates a new thread to handle the request, even if other worker threads are idle. If there are more than **minPoolSize** but less than **maxPoolSize** threads running, the system allocates a new thread only if the queue is full (as expressed by the actual queue size).
- If the pool currently has more than **minPoolSize** threads, excess threads will be terminated if they have been idle for more than **keepAliveTime**. This provides a means of reducing resource consumption when the pool is not actively being used. If the pool becomes more active later, new threads will be allocated. Note that the keep-alive policy applies only when there are more than **minPoolSize** threads.

As a result of this behavior, the system rejects a request when the queue is full and the actual pool size equals **maxPoolSize**.

8.9.5.2 CLI Context

Following is an example of configuring an OCI-C executor (for both queue and thread pool).

```
PS_CLI/System/CommunicationUtility/DefaultSettings/Executors/OCIC/Queue> help
set
The command is used to modify Queue settings.

Parameters description:
attribute: The name of an attribute to modify.
capacity : This parameter specifies the maximum number of requests that can
be
```

```
            queued while waiting for other tasks to complete during execution.

======================================================================
set
    <attribute>, Multiple Choice = {capacity}
        <capacity>, Integer {1 to 2147483647}

PS_CLI/System/CommunicationUtility/DefaultSettings/Executors/OCIC/Queue> set
capacity 25000
*** Warning: BroadWorks needs to be restarted for the changes to take effect
***

PS_CLI/System/CommunicationUtility/DefaultSettings/Executors/OCIC/Queue> get
  capacity = 25000

PS_CLI/System/CommunicationUtility/DefaultSettings/Executors/OCIC/Queue>
```

```
PS_CLI/System/CommunicationUtility/DefaultSettings/Executors/OCIC/ThreadPool> help
set
The command is used to modify ThreadPool settings.

Parameters description:
attribute    : The name of an attribute to modify.
min          : This parameter specifies the minimum number of threads to keep
               in steady state. Lower thread counts are possible during
               initialization.
max          : This parameter specifies the maximum number of threads this
pool
               can contain.
keepALiveTime: This parameter specifies the amount of idle time before an
               excess thread is terminated.

======================================================================
set
    <attribute>, Multiple Choice = {min, max, keepALiveTime}
        <min>, Integer {1 to 10000}
        <max>, Integer {1 to 10000}
        <keepALiveTime>, Integer {1 to 3600}

PS_CLI/System/CommunicationUtility/DefaultSettings/Executors/OCIC/ThreadPool> set
min 2 max 5 keepALiveTime 30
*** Warning: Broadworks needs to be restarted for the changes to take effect
***

PS_CLI/System/CommunicationUtility/DefaultSettings/Executors/OCIC/ThreadPool> get
  min = 2
  max = 5
  keepALiveTime = 30

PS_CLI/System/CommunicationUtility/DefaultSettings/Executors/OCIC/ThreadPool>
```

8.9.6 External Authentication Agent Support

The Profile Server can be configured to recognize authentication tokens provided by an external authentication agent as described in the *BroadWorks External Portal Integration Guide Developer's Guide* [6]. The *allowExtAuthAgent* attribute controls recognition of the tokens (provided using *HTTP* headers).

8.9.7 Name Service

The Profile Server integrates the BroadWorks Name Service into the BWCommunicationUtility. This behavior is the same as the Open Client Server (OCS) behavior. The webapps also benefit from this mechanism since they go through the BroadWorks Name Service for name resolution.

When there is a requirement to use SRV lookup for the **nslocation** service, an operator has to enable the SRV lookup parameter.

When enabled, the NSclusterAddress is taken as a domain name for a SRV lookup to the **nslocation** service on the **tcp** protocol: **_nslocation._tcp.<clusterAddress>**. If the SRV lookup fails, an A lookup is then performed with the clusterAddress. The Profile Server's NameService can be configured with proper SRV records, including weight and cost, for the nslocation service.

For details about configuring the BroadWorks Name Service, see the *BroadWorks Name Service Integration Feature Description* [11].

Following is an example of enabling the SRV lookup.

```
PS_CLI/System/CommunicationUtility/DefaultSettings> help set
This command is used to modify BroadWorks communication utility attributes in
the system.  The mode attribute (AS or NS) refers to the integration scheme
used to integrate with the Application Server clusters.  In Network Server
(NS)
mode, Network Server lookups are used to dynamically resolve the Application
Server clusters.  The Application Server (AS) mode allows direct integration
with a single Application Server cluster.
The transaction limit rates are computed as follows:
- Global - globalTransactionLimit divided by the transactionLimitPeriodSecs
- per User - userTransactionLimit divided by the transactionLimitPeriodSecs
Note that all Application Server clusters must use the same OCI port when
using
the Network Server (NS) mode.
++++noFor your changes to take effect, you must restart the system.++++nc

Parameters description:
attribute                : Additional attributes to modify specific to the
NS
                           mode.
mode                     : Select the mode, which is either AS for an
                           Application Server or NS for a Network Server.
                           The attributes applicable to each mode is shown
                           below.
AS                       : The CommunicationUtility operating in AS mode.
asPrimaryAddress         : The primary Application Server's IP address,
host,
                           or domain.
asSecondaryAddress       : The secondary Application Server's IP address,
                           host, or domain.
asOCIPort                : Application Server OCI-P (to the Provisioning
Server
                           process) plain tcp/ip listening port.
asOCISecurePort          : Application Server OCI-P (to the Provisioning
Server
                           process) plain secure tcp/ip listening port.
asOCICPort               : Application Server OCI-C (to the eXecution Server
                           process) plain tcp/ip listening port.
asOCICSecurePort         : Aplication Server OCI-C (to the eXecution Server
                           process) plain secure tcp/ip listening port.
NS                       : The CommunicationUtility operating in NS mode.
nsSeedClusterAddress     : The Network Server's IP address, host, or domain.
locationApiTimeoutSecs   : Sets the Location API response time-out (in
                           seconds).
enableSrvLookup          : Enables or disable SRV lookup for NS location API
```

```
                              requests.
provisionOnSecondary      : Allows or disallows provisioning on the secondary
                              Application Server.
reconnectionTimerSecs     : Sets the reconnection timer (in seconds).
responseTimeoutSecs       : Sets the response timeout (in seconds).
userTransactionLimit      : Sets the user transaction limit.
transactionLimitPeriodSecs: Sets the transaction limit period (in seconds).
useSecureBCCT             : Sets if secure BCCT is used.

======================================================================
set
    <attribute>, Multiple Choice = {mode, provisionOnSecondary,
reconnectionTimerSecs, responseTimeoutSecs, userTransactionLimit,
transactionLimitPeriodSecs, useSecureBCCT}
        <mode>, Choice = {AS, NS}
            AS:
                <asPrimaryAddress>, IP address | host | domain (1 to 80
chars)
                [<attribute>, Multiple Choice = {asSecondaryAddress,
asOCIPort, asOCISecurePort, asOCICPort, asOCICSecurePort}]
                    <asSecondaryAddress>, IP address | host | domain (1 to 80
chars)
                    <asOCIPort>, Integer {1024 to 65535}
                    <asOCISecurePort>, Integer {1024 to 65535}
                    <asOCICPort>, Integer {1024 to 65535}
                    <asOCICSecurePort>, Integer {1024 to 65535}
            NS:
                <nsSeedClusterAddress>, IP address | host | domain (1 to 80
chars)
                [<attribute>, Multiple Choice = {asOCIPort, asOCISecurePort,
asOCICPort, asOCICSecurePort, locationApiTimeoutSecs, enableSrvLookup}]
                    <asOCIPort>, Integer {1024 to 65535}
                    <asOCISecurePort>, Integer {1024 to 65535}
                    <asOCICPort>, Integer {1024 to 65535}
                    <asOCICSecurePort>, Integer {1024 to 65535}
                    <locationApiTimeoutSecs>, Integer {1 to 10}
                    <enableSrvLookup>, Choice = {false, true}
        <provisionOnSecondary>, Choice = {false, true}
        <reconnectionTimerSecs>, Integer {1 to 300}
        <responseTimeoutSecs>, Integer {1 to 60}
        <userTransactionLimit>, Integer {1 to 65535}
        <transactionLimitPeriodSecs>, Integer {1 to 300}
        <useSecureBCCT>, Choice = {false, true}

PS_CLI/System/CommunicationUtility/DefaultSettings>
PS_CLI/System/CommunicationUtility/DefaultSettings> set mode NS nsSrvLocation
enableSrvLookup true
*** Warning: Broadworks needs to be restarted for the changes to take effect
***

PS_CLI/System/CommunicationUtility/DefaultSettings> get
NS:
  nsSeedClusterAddress= nsSeedClusterSrvExample
  asOCIPort=2220
  asOCISecurePort=2320
  asOCICPort=2221
  asOCICSecurePort=2321
  locationApiTimeoutSecs=8
  enableSrvLookup=true
provisionOnSecondary=false
reconnectionTimerSecs=30
responseTimeoutSecs=10
userTransactionLimit=5
transactionLimitPeriodSecs=1
useSecureBCCT=false
```

```
PS_CLI/System/CommunicationUtility/DefaultSettings
```

Following is an example of configuring the *namedefs* configuration file.

```
Example configuration in namedefs:
    Assuming namedefs include the entry described below, CommPilot uses
    this entry for mapping the nsSeedClusterAddress.

      Example configuration in /usr/local/broadworks/bw_base/conf/namedefs:
 _nslocation._tcp.nsSeedClusterSrvExample SRV 1 5 5060 nsSeedClusterExample1
 _nslocation._tcp.nsSeedClusterSrvExample SRV 1 2 5060 nsSeedClusterExample2

 nsSeedClusterAExample IN A 192.168.13.183
 nsSeedClusterExample1 IN A 192.168.13.184
 nsSeedClusterExample2 IN A 192.168.13.185
```

8.10 Configuration of IMS Components

The following applications are required when the Profile Server is used in the context of IMS architecture:

- Provisioning
- MediaFiles
- DeviceManagementFiles
- OpenClientServer
- DBSObserver

8.10.1 Configuration of Provisioning Application

The Provisioning Application is responsible for providing provisioning interfaces (man and machine) for the BroadWorks transparent data. In addition to exposing various APIs, it also exposes a system and group administrator portal along with an end-user portal. For more information on this application, see the *BroadWorks IMS Configuration Guide* [4].

8.10.2 Configuration of Media Files Application

The Media Files Repository application is the centralized repository for all media files. This function is provided by a web application deployed in Tomcat. The media files repository settings are configurable through the CLI, under the *PS_CLI/Applications/BroadworksFileRepos* level.

8.10.2.1 MediaFiles Web Application Management

The Media Files Repository application can be deployed or undeployed on any level. By default, the media files repository web application is deployed on the */media* level. This can be configured differently through the CLI, under the *Maintenance/ManagedObjects* level.

8.10.2.2 Root Directory

The Media Files Repository application level is configured as part of the web application deployment. This level is specified in file repository requests coming from Application Servers and Execution Server. The administrator can also provision a system-wide root directory under which all the files are persisted. The mapping between a file repository request and the actual resource starts with an incoming request such as:

http://fileReposFQDN:portNumber/fileReposRootDirInAppServerContextPart/fileReposRootDirInAppServerRemainingPart/deviceTypePath/fileResource

fileReposRootDirInAppServerContextPart is the File Repository web application level name. The web application services this request. The resource file referenced ends up being:

/rootDirectory/fileReposRootDirInAppServerRemainingPart/deviceTypePath/fileResource

… where the *rootDirectory* is the new system-wide file repository root directory defined on the Profile Server.

8.10.2.3 User Names

The Media Files Repository application supports a mode in which HTTP file requests are authenticated using a user name/password technique. Authentication is configurable and is recommended. The user names and password are configured using the CLI.

8.10.3 Configuration of Device Management Files Application

The Device Management Files application is the centralized repository for all the device type files. This function is provided by a web application deployed in Tomcat. The Device Management files repository settings are configurable through the CLI under the *PS_CLI/Applications/DeviceManagementFiles* level.

8.10.3.1 Device Management Files Web Application Management

The Device Management Files Repository application can be deployed or undeployed on any level. By default, the Device Management files repository web application is deployed on the */DeviceManagement* level. This can be configured differently through the CLI under the *Maintenance/ManagedObjects* level.

8.10.3.2 User Names

The Device Management Files Repository application supports a mode in which HTTP file requests are authenticated using a user name/password technique. Authentication is always enabled. The user names and password are configured using the CLI.

8.10.4 Configuration of Open Client Server Application

The Open Client Server application is used to proxy OCI-P, Client Application Protocol (CAP), Network Server Operations Support System (NSOSS), and Network Server Open Client Interface (NSOCI) between client and cluster of Application Servers and Network Servers. The Open Client Server settings are configurable through the CLI, under the *PS_CLI/Applications/OpenClientServer* level.

8.10.5 Configuration of Database Observer Application

The Database Observer application continuously monitors the primary and target standby databases and evaluates whether a failover is necessary. Then it initiates a fast-start failover when conditions warrant.

8.11 Configuration of Device Management Components

The BroadWorksFileRepos application is required to be deployed for Device Management. Optionally, the BroadworksFileReposExtdCapture could be deployed to support extended file capture.

8.11.1 Configuration of BroadWorksFileRepos Application

The file repository function of the Profile Server is provided by a web application deployed in Tomcat. The file repository settings are configurable through the CLI under the *PS_CLI/Applications/BroadWorksFileRepos* level.

8.11.1.1 FileRepos Web Application Management

The File Repository application can be deployed or undeployed on any level. By default, the file repository web application is deployed on the root ('/') level. This can be configured differently through the CLI under the *Maintenance/ManagedObjects* level.

8.11.1.2 Root Directory

The File Repository application level is configured as part of the web application deployment. This level is specified in file repository requests coming from Application Servers and Xtended Services Platforms. The administrator can also provision a system-wide root directory under which all the files are persisted. The mapping between a file repository request and the actual resource starts with an incoming request such as:

http://fileReposFQDN:portNumber/fileReposRootDirInAppServerContextPart/fileReposRootDirInAppServerRemainingPart/deviceTypePath/fileResource

fileReposRootDirInAppServerContextPart is the File Repository web application level name. The web application services this request. The resource file referenced ends up being:

/rootDirectory/fileReposRootDirInAppServerRemainingPart/deviceTypePath/fileResource

… where the *rootDirectory* is the new system-wide file repository root directory defined in the Profile Server.

8.11.1.3 Access List

The File Repository application maintains an access list to control all the IP addresses that are allowed to use HTTP services on the server. The access list is configured using the CLI (under the CLI level *Applications/BroadworksFileRepos/NetworkAccessLists/WebDav*). Typically, Application Servers and Xtended Services Platform IP addresses (private) should be on the access list. The access list is persisted in a flat file replicated using rsync to other Profile Servers in the farm.

8.11.1.4 User Names

The File Repository application supports a mode in which HTTP file requests are authenticated using a user name/password technique. This is not the user names that access devices use to access BroadWorks; these are user names that the Application Server and Xtended Services Platform use to retrieve or deposit files on the Profile Server. Authentication is configurable and is recommended. The user names and passwords are configured using the CLI.

8.11.1.5 Encryption

The File Repository web application can store files in an encrypted fashion. The use of encryption is configurable by setting the *encryptionType* parameter (RC4, AES, none).

When the encryption is enabled for the BroadworksFileRepos web application running on the Profile Server, any file received in a file creation/update request is encrypted before being stored to disk. However, when encryption is disabled, any file received in a file creation/update request is processed normally without any encryption.

8.11.1.6 Extended File System

The Profile Server directory structure can be extended to allow for the storage of more than 32000 subdirectories, a value inherited from a file system limitation. When enabled, the extended file system provides the capability for a directory to hold more than 32,000 (32k) subdirectories and allows for the storage and management of potentially millions of subdirectories under a single directory. There is no impact on the natural structure of the directory until a configurable maximum size is reached. It is only at this time that the virtual directories are created to store the files that exceed the maximum size allowed. These virtual directories are referenced as buckets.

The following parameters are available:

- *enabled* – This parameter specifies if the directory enhancements are enabled.
- *fileBucketCapacity* – This parameter specifies the number of files per bucket. If left unspecified (or cleared), the file bucket functionality is disabled.
- *cacheCapacity* – This parameter specifies the maximum number of cached entries. If the value is unspecified (or cleared), the cache loads the maximum number of entries with the available memory.
- *mibPrefix* – This parameter specifies the Management Information Base (MIB) prefix.

8.11.1.7 File Replication

The File Replication is performed using HTTP. The following parameters are available:

- *username* – This is the user name used for the replication authentication.
- *password* – This is the password used for the replication authentication.
- *retryInterval* – This is the parameter that specifies the interval between each replication attempt.
- *failedAttemptsBeforeAlarm* – This is the number of failed attempts before the bwFileReposReplicationFailure alarm is raised.
- *timeout* – This is the maximum duration of a request, after which the request times out and the replication attempt is marked as failed.
- *Replicator* – The replicator executes the replication tasks and allows the user to configure the following:
  - *queue capacity*: This parameter specifies the maximum queue capacity. If left unspecified (or cleared), it indicates unlimited queuing.
  - *sizeThreshold*: This parameter specifies the queue size limits that control alarm generation.
  - *latencyThreshold*: This parameter specifies the queue latency limits that control alarm generation.
- *capacity* – This parameter overrides the replicator basic executor queue capacity.

8.11.2 Configuration of BroadWorksFileReposExtCapture Application

The extended capture file repository has the same functionality as the BroadWorks file repository, with the addition of the storage management function. Files under this repository are deleted when they are older than a configurable number of days. In addition, the number of files within a directory is limited to a configurable amount.

8.11.2.1 FileRepos Web Application Management

The Extended Capture File Repository application can be deployed or undeployed on any level.

8.11.2.2 Root Directory

The Extended Capture File Repository application level is configured as part of the web application deployment. This level is specified in file repository requests coming from Application Servers and Xtended Services Platforms. The administrator can also provision a system-wide root directory under which all the files are persisted. The mapping between a file repository request and the actual resource starts with an incoming request such as:

http://fileReposFQDN:portNumber/fileReposExtdCaptureRootDirInAppServerContextPart/fileReposExtdCaptureRootDirInAppServerRemainingPart/deviceTypePath/fileResource

fileReposExtdCaptureRootDirInAppServerContextPart is the Extended Capture File Repository web application level name. The web application services this request. The resource file referenced ends up being:

/rootDirectory/fileReposExtdCaptureRootDirInAppServerRemainingPart/deviceTypePath/fileResource

… where the rootDirectory is the new system-wide extended capture file repository root directory defined in the Profile Server.

8.11.2.3 Access List

The Extended Capture File Repository application maintains an access list to control all the IP addresses that are allowed to use HTTP services on the server. The access list is configured using the CLI (under the CLI level *Applications/BroadworksFileReposExtdCapture/NetworkAccessLists/WebDav*). Typically, Application Servers and Xtended Services Platform IP addresses (private) should be on the access list. The access list is persisted in a flat file replicated using rsync to other Profile Servers in the farm.

8.11.2.4 User Names

The Extended Capture File Repository application supports a mode in which HTTP file requests are authenticated using a user name/password technique. This is not the user names that access devices use to access BroadWorks; these are user names that the Application Server and Xtended Services Platform use to retrieve or deposit files on the Profile Server. Authentication is configurable and is recommended. The user names and passwords are configured using the CLI.

8.11.2.5 Extended File System

The Profile Server directory structure can be extended to allow for the storage of more than 32000 subdirectories, a value inherited from a file system limitation. When enabled, the extended file system provides the capability for a directory to hold more than 32,000 (32k) subdirectories and allows for the storage and management of potentially millions of subdirectories under a single directory. There is no impact on the natural structure of the directory until a configurable maximum size is reached. It is only at this time that the virtual directories are created to store the files that exceed the maximum size allowed. These virtual directories are referenced as “buckets”.

The following parameters are available:

- *enabled* – This parameter specifies if the directory enhancements are enabled.
- *fileBucketCapacity* – This parameter specifies the number of files per bucket. If left unspecified (or cleared), the file bucket functionality is disabled.
- *cacheCapacity* – This parameter specifies the maximum number of cached entries. If the value is unspecified (or cleared), the cache loads the maximum number of entries with the available memory.
- *mibPrefix* – This parameter specifies the Management Information Base (MIB) prefix.

8.11.2.6 File Replication

File Replication is performed using HTTP. The following parameters are available:

- *username* – This is the user name used for the replication authentication.
- *password* – This is the password used for the replication authentication.
- *retryInterval* – This is the parameter that specifies the interval between each replication attempt.
- *failedAttemptsBeforeAlarm* – This is the number of failed attempts before the bwFileReposReplicationFailure alarm is raised.
- *timeout* – This is the maximum duration of a request, after which the request times out and the replication attempt is marked as failed.
- *Replicator* – The replicator executes the replication tasks and allows the user to configure the following:
  - *queue capacity* – This parameter specifies the maximum queue capacity. If left unspecified (or cleared), it indicates unlimited queuing.
  - *sizeThreshold* – This parameter specifies the queue size limits that control alarm generation.
  - *latencyThreshold* – This parameter specifies the queue latency limits that control alarm generation.
- *capacity* – This parameter overrides the replicator basic executor queue capacity.

8.11.2.7 Storage Management

Files stored in the Extended Capture File Repository are deleted after a certain time. A task is run on a daily basis to perform those tasks. The time at which the task is executed and the number of days after which a file is to be deleted is configurable via the CLI.

The number of files per directory stored in the Extended Capture File Repository is limited. Whenever the number of files in a directory is greater than the maximum configured, the oldest files are deleted until the maximum number of files is reached. This cleanup is performed by a concurrent collector that continuously performs the required cleanup as files are added to the repository. The concurrent collector includes a thread fronted by a queue, which holds cleanup requests for directories in which files have been added. Since the cleanup occurs asynchronously to the HTTP operations, it is possible for the number of files in some directories to briefly exceed the configured limit. This situation should be resolved quickly when the queue is emptied. If the concurrent collector does not keep up with the incoming flow, the queue starts filling. The queue occupancy is monitored and an alarm is generated if a configurable threshold (x %) is reached. When the queue is full, additional cleanup requests are dropped. When this occurs, a log is generated. If requests are dropped, a forced cleanup can be performed through the CLI. The CLI tool can be invoked from *StorageManagement/Tools/cleanupFileSystem*.

The maximum number of files for each directory is configurable under *StorageManagement/Root*. This setting applies to all directories within the repository. The executor performing the cleanup is configurable under *StorageManagement/ConcurrentCollector*. The system administrator has the ability to modify the capacity of the queue as well as the queue size threshold at which alarms are generated or cleared.

8.12 Configuration of Centralized Audit Logs components

The *LogRepository* application is required to be deployed for Centralized Audit Logs.

8.12.1 Configuration of LogRepository Application

The Log Repository application is the centralized repository for all audit logs. This function is provided by a web application deployed in Tomcat. The log repository settings are configurable through the CLI, under the *PS_CLI/Applications/LogRepository* level.

8.12.1.1 LogRepository Web Application Management

The Log Repository application can be deployed or undeployed on any level. By default, the log repository web application is deployed on the */logrepos* level. This can be configured differently through the CLI, under the *Maintenance/ManagedObjects* level.

8.12.1.2 Root Directory

The Log Repository application level is configured as part of the web application deployment. This level is specified in file repository requests coming from any server. The administrator can also provision a system-wide root directory under which all the files are persisted. The mapping between a file repository request and the actual resource starts with an incoming request such as:

http://logReposFQDN:portNumber/logReposRootDirContextPart/logReposRootDirRemainingPart/fileResource

logReposRootDirContextPart is the Log Repository web application level name. The web application services this request. The resource file referenced ends up being:

/rootDirectory/logReposRootDirRemainingPart/fileResource

… where the rootDirectory is the new system-wide file repository root directory defined on the Profile Server.

8.12.1.3 User Names

The Log Repository application supports a mode in which HTTP file requests are authenticated using a user name/password technique. Authentication is configurable and is recommended. The user names and password are configured using the CLI.

Storage Management Files stored in the Log Repository are deleted and/or compressed after a certain time. A task is run on a daily basis to perform those tasks. The time at which the task is executed and the number of days after which a file is to be deleted and/or compressed is configurable via the CLI.

8.13 Configuration of Meet-Me Audio Conferencing Components

The Meet-Me Conferencing Repository application is required to be deployed for Meet-Me Audio Conferencing.

8.13.1 Configuration of Meet-Me Conferencing Repository Application

The Meet-Me Conferencing Repository application is the centralized repository for all the recordings. The Meet-Me Conferencing repository settings are configurable through the CLI under the *PS_CLI/Applications/MeetMeConferencingRepository* level.

8.13.1.1 Meet-Me Conferencing Repository Web Application Management

The Meet-Me Conferencing Repository application can be deployed/undeployed on any level. By default, the Meet-Me Conferencing repository web application is deployed on the */MeetMeConfRepos* level. This can be configured differently through the CLI under the *Maintenance/ManagedObjects* level.

8.13.1.2 Root Directory

The Meet-Me Conferencing Repository application level is configured as part of the web application deployment. This level is specified in recording requests coming from Media Servers and Application Servers. The administrator can also provision a system-wide root directory under which all the files are persisted. The mapping between a recording request and the actual resource starts with an incoming request such as:

http://meetMeConferencingReposFQDN:portNumber/meetMeConferencingReposRootDirInAppServerContextPart/meetMeConferencingReposRootDirInAppServerRemainingPart/fileResource

meetMeConferencingReposRootDirInAppServerContextPart is the Meet-Me Conferencing Repository web application level name. The web application services this request. The resource file referenced ends up being:

/rootDirectory/meetMeConferencingReposRootDirInAppServerRemainingPart/fileResource

… where the *rootDirectory* is the new system-wide file repository root directory defined in the Profile Server.

8.13.1.3 Access List

The Meet-Me Conferencing Repository application maintains an access list to control all the IP addresses that are allowed to use HTTP services on the server. The access list is configured using the CLI (under the CLI level *Applications/MeetMeConferencingRepository /NetworkAccessLists/WebDav*). Typically, Application Servers and Media Servers IP addresses (private) should be on the access list. The access list is persisted in a flat file replicated using rsync to other Profile Servers in the farm.

8.13.1.4 Storage Management

Files stored in the Meet-Me Conferencing Repository are deleted after a certain time. A task is run on a daily basis to perform this task. The time at which the task is executed and the number of days after which a file is to be deleted is configurable via the CLI.

8.14 Configuration of Enhanced Call Center Components

The following applications are required when the Profile Server is used in the context of Enhanced Call Center:

- CCReporting
- CCReportingDBManagement
- CCReportingRepository

For more information on these applications, see *Call Center Solution Guide* [7].

8.14.1 Configuration of Call Center Reporting Application

The Call Center Reporting application implements the reporting engine, which retrieves data from the centralized database and generates the report. The Call Center Reporting application settings are configurable through the CLI, under the *PS_CLI/Applications/CCReporting* level.

8.14.2 Configuration of Call Center Reporting DB Management Application

This Call Center Reporting DB Management application manages the Enhanced Call Center Reporting datastore (BWECCR schema) on the Database Server. The BWECCR schema is created when the application is first deployed. The Call Center Reporting DB Management application settings are configurable through the CLI, under the *PS_CLI/Applications/CCReportingDBManagement* level.

8.14.3 Configuration of Call Center Reporting Repository Application

The Call Center Reporting Repository application is the centralized repository for all the enhanced call center files. The Call Center Reporting repository settings are configurable through the CLI under the *PS_CLI/Applications/CCReportingRepository* level.

8.14.3.1 Call Center Reporting Repository Web Application Management

The Call Center Reporting Repository application can be deployed or undeployed on any level. By default, the Call Center Reporting repository web application is deployed on the */CCReportingRepository* level. This can be configured differently through the CLI under the *Maintenance/ManagedObjects* level.

8.14.3.2 Root Directory

The Call Center Reporting Repository application level is configured as part of the web application deployment. This level is specified in call center repository requests coming from Application Servers and Profile Servers. The administrator can also provision a system-wide root directory under which all the files are persisted. The mapping between a call center repository request and the actual resource starts with an incoming request such as:

http://ccReportingReposFQDN:portNumber/ccReportingReposRootDirInAppServerContextPart/ccReportingReposRootDirInAppServerRemainingPart/fileResource

ccReportingReposRootDirInAppServerContextPart is the Call Center Repository web application level name. The web application services this request. The resource file referenced ends up being:

/rootDirectory/ccReportingReposRootDirInAppServerRemainingPart/fileResource

… where the rootDirectory is the new system-wide file repository root directory defined in the Profile Server.

8.14.3.3 Access List

The Call Center Repository application maintains an access list to control all the IP addresses that are allowed to use HTTP services on the server. The access list is configured using the CLI (under CLI level *Applications/CCReportingRepository/NetworkAccessLists/WebDav*). Typically, Application Servers and Profile Servers IP addresses (private) should be on the access list. The access list is persisted in a flat file replicated using rsync to other Profile Servers in the farm.

8.14.3.4 Cleanup

Files stored in the Call Center Reporting Repository (under */DataArchive* and */ReportArchive*) are deleted and or compressed after a certain time. A task is run on a daily basis to perform those tasks. The time at which the task is executed and the number of days after which a file is to be deleted and/or compressed is configurable via the CLI.

8.15 Configuration of Enhanced Call Log Components

The following applications are required when the Profile Server is used in the context of Enhanced Call Log:

- EnhancedCallLogsDBManagement
- ECLQuery
- ECLReportingRepository

For more information on this application, see the *BroadWorks Enhanced Call Logs Solution Guide* [8].

8.15.1 Configuration of EnhancedCallLogsDBManagement Application

The Enhanced Call Logs DB Management application manages the Enhanced Call Logs data store (BWECL schemas) on the Database Server. Every instance of the BWECL schema should be defined on a single Profile Server and should be created by issuing commands from that Profile Server. The Enhanced Call Logs DB Management application settings are configurable through the CLI, under the *PS_CLI/Applications/EnhancedCallLogsDBManagement* level.

8.15.2 Configuration of Enhanced Call Log Query Application

The Enhanced Call Logs Query web application provides a private interface for the Public Enhanced Call Logs Query web application to query the enterprise or group enhanced call logs data. The Enhanced Call Logs Query settings are configurable through the CLI, under the *PS_CLI/Applications/ECLQuery* level.

8.15.3 Configuration of Enhanced Call Log Repository Application

The Enhanced Call Logs repository application is the centralized repository for all the enhanced call logs files. The Enhanced Call Logs repository settings are configurable through the CLI under the *PS_CLI/Applications/ECLReportingRepository* level.

8.15.3.1 Call Center Reporting Repository Web Application Management

The Enhanced Call Logs repository application can be deployed or undeployed on any level. By default, the Enhanced Call Logs repository web application is deployed on the */ECLReportingRepository* level. This can be configured differently through the CLI under the *Maintenance/ManagedObjects* level.

8.15.3.2 Root Directory

The Enhanced Call Logs repository application level is configured as part of the web application deployment. This level is specified in enhanced call log repository requests coming from Xtended Services Platform and Profile Servers. The administrator can also provision a system-wide root directory under which all the files are persisted. The mapping between an enhanced call log repository request and the actual resource starts with an incoming request such as:

http://eclReportingReposFQDN:portNumber/eclReportingReposRootDirInXspContextPart/eclReportingReposRootDirInAppServerRemainingPart/fileResource

eclReportingReposRootDirInXspContextPart is the Enhanced Call Log repository web application level name. The web application services this request. The resource file referenced ends up being:

/rootDirectory/eclReportingReposRootDirInXspRemainingPart/fileResource

… where the *rootDirectory* is the new system-wide file repository root directory defined in the Profile Server.

8.15.3.3 Access List

The Enhanced Call Logs repository application maintains an access list to control all the IP addresses that are allowed to use HTTP services on the server. The access list is configured using the CLI (under CLI level *Applications/ECLReportingRepository/NetworkAccessLists/WebDav*). Typically, Xtended Services Platform and Profile Servers IP addresses (private) should be on the access list. The access list is persisted in a flat file replicated using rsync to other Profile Servers in the farm.

8.15.3.4 Cleanup

Files stored in the Enhanced Call Logs repository (under */ECLQueryData*) are deleted after a certain time. A task is run on a daily basis to perform those tasks. The time at which the task is executed and the number of days after which a file is to be deleted is configurable via the CLI.

8.15.3.5 File Replication

File Replication is performed using HTTP. The following parameters are available:

- *username* – This is the user name used for the replication authentication.
- *password* – This is the password used for the replication authentication.
- *retryInterval* – This is the parameter that specifies the interval between each replication attempt.
- *failedAttemptsBeforeAlarm* – This is the number of failed attempts before the bwFileReposReplicationFailure alarm is raised.
- *timeout* – This is the maximum duration of a request, after which the request times out and the replication attempt is marked as failed.
- *Replicator* – The replicator executes the replication tasks and allows the user to configure the following:
  - *queue capacity* – This parameter specifies the maximum queue capacity. If left unspecified (or cleared), it indicates unlimited queuing.
  - *sizeThreshold* – This parameter specifies the queue size limits that control alarm generation.
  - *latencyThreshold* – This parameter specifies the queue latency limits that control alarm generation.
- *capacity* – This parameter overrides the replicator basic executor queue capacity.

8.16 Configuration of Instant Messaging and Presence Components

The following applications are required when the Profile Server is used in the context of Instant Messaging and Presence:

- MessageArchive

8.16.1 Configuration of MessageArchive Application

The Message Archive repository function of the Profile Server is provided by a web application deployed in Tomcat. The Message Archive repository settings are configurable through the CLI under the *PS_CLI/Applications/MessageArchive* level.

8.16.1.1 MessageArchive Web Application Management

The Message Archive repository application can be deployed or undeployed on any level. By default, the message archive repository web application is deployed on the */MessageArchive* level. This can be configured differently through the CLI under the *Maintenance/ManagedObjects* level.

8.16.1.2 Root Directory

The Message Archive repository application level is configured as part of the web application deployment. This level is specified in message archive repository requests coming from Messaging Servers (UMS). The administrator can also provision a system-wide root directory under which all the files are persisted. The mapping between a message archive repository request and the actual resource starts with an incoming request such as:

http://fileReposFQDN:portNumber/fileReposRootDirInMessagingServerContextPart/fileReposRootDirInMessagingServerRemainingPart/deviceTypePath/fileResource

fileReposRootDirInMessagingServerContextPart is the Message Archive repository web application level name. The web application services this request. The resource file referenced ends up being:

/rootDirectory/fileReposRootDirInMessagingServerRemainingPart/deviceTypePath/fileResource

… where the rootDirectory is the new system-wide Message Archive repository root directory defined in the Profile Server.

8.16.1.3 Access List

The Message Archive repository application maintains an access list to control all the IP addresses that are allowed to use HTTP services on the server. The access list is configured using the CLI (under the CLI level *Applications/MessageArchive/NetworkAccessLists/WebDav*). Typically, Messaging Servers should be on the access list. The access list is persisted in a flat file replicated using rsync to other Profile Servers in the farm.

8.16.1.4 User Names

The Message Archive Repository application supports a mode in which HTTP file requests are authenticated using a user name/password technique. This is not the user names that access devices use to access BroadWorks; these are user names that the Messaging Server uses to retrieve or deposit files on the Profile Server. Authentication is configurable and is recommended. The user names and passwords are configured using the CLI.

8.16.1.5 Encryption

The Message Archive repository web application can store files in an encrypted fashion. The use of encryption is configurable by setting the *encryptionType* parameter (RC4, AES, or none).

When the encryption is enabled for the Message Archive web application running on the Profile Server, any file received in a file creation/update request is encrypted before being stored to disk. However, when encryption is disabled, any file received in a file creation/update request is processed normally without any encryption.

8.16.1.6 File Replication

File Replication is performed using HTTP. The following parameters are available:

- *username* – This is the user name used for the replication authentication.
- *password* – This is the password used for the replication authentication.
- *retryInterval* – This is the parameter that specifies the interval between each replication attempt.

- *failedAttemptsBeforeAlarm* – This is the number of failed attempts before the bwFileReposReplicationFailure alarm is raised.
- *timeout* – This is the maximum duration of a request, after which the request times out and the replication attempt is marked as failed.
- *Replicator* – The replicator executes the replication tasks and allows the user to configure the following:
  - *queue capacity* – This parameter specifies the maximum queue capacity. If left unspecified (or cleared), it indicates unlimited queuing.
  - *sizeThreshold* – This parameter specifies the queue size limits that control alarm generation.
  - *latencyThreshold* – This parameter specifies the queue latency limits that control alarm generation.
- *capacity* – This parameter overrides the replicator basic executor queue capacity.

8.16.1.7 Storage Management

Files stored in the Message Archive repository are deleted after a certain time. A task is run on a daily basis to perform those tasks. The time at which the task is executed and the number of days after which a file is to be deleted is configurable via the CLI.

9 Network Configuration Changes

This section provides information on the Profile Server configuration operations required when the IP or host name of the Profile Server, Application Server, or Network Server is changed.

9.1 Change IP or Host Name of Profile Server

When changing the IP address or host name of the Profile Server, the HTTP server, and in some cases the HTTP Alias configuration, must be updated.

The overall procedure for updating the IP address or host name of a BroadWorks server is described in the *BroadWorks Maintenance Guide* [5]. This section details the operations specific to the Profile Server.

After the IP address or host name has been updated at the operating system (OS) level, the HTTP server configuration must be updated.

9.1.1 Update HTTP Server Configuration

Updating the HTTP server configuration is done using the CLI, under the *PS_CLI/Interface/Http/HttpServer* level.

For IP address changes:

1) First, for each HTTP server using the old IP, create a new HTTP server on the new IP with the same attributes. Use the add command (for example, add 192.168.12.1 80 host false).
2) When all new HTTP servers are created, simply delete those using the old IP. Use the delete command (for example, delete 192.168.13.1 80).

For host name changes:

1) If the host name was being used by an HTTP server, update the HTTP server name with the new host name. Use the set command (for example, set 192.168.12.1 80 name=newhost).
2) If using the host name for a secured HTTP server, the Profile Server generates self-signed certificates. These certificates must be signed. For more information, see section *8.7.5 Manage SSL Certificates*.

9.2 Change IP or Host Name of Network Server

The BWCommunicationUtility must be reconfigured with the new IP or host name if the integration mode is set to “NS”. If it is set to “NS”, the IP address or host name of the Network Server must be entered using the CLI, under the *PS_CLI/System/CommunicationUtility/DefaultSettings* level. For more information, see section *8.9.1 Integration Mode*.

Change the *nsSeedClusterAddress* parameter to the new value.

If using a clusterFQDN to reach the Network Server(s), then this step is only required when the FQDN changes.

9.3 Change IP or Host Name of Application Server

The BWCommunicationUtility must be reconfigured with the new IP or host name if the integration mode is set to “AS”. If it is set to “AS”, the IP address or host name of the Application Server must be entered using the CLI under the *PS_CLI/System/CommunicationUtility/DefaultSettings* level. For more information, see section *8.9.1 Integration Mode*.

Change the *asPrimaryAddress* or the *asSecondaryAddress* as required.

10 Troubleshooting

This section provides general information about Profile Server troubleshooting.

10.1 Logs

Most Profile Server components issue logs. The following subsections provide information on where logging information can be found on the Profile Server.

10.1.1 Apache Logs

The Apache server used by the Profile Server issues logs to the */var/broadworks/logs/apache* directory.

10.1.2 Tomcat Logs

The Tomcat servlet container used by the Profile Server issues logs to the */var/broadworks/logs/tomcat* directory. This log contains the container internal logs and possibly the exceptions it encounters while executing the various web applications.

10.1.3 BroadWorks Application Logs

BroadWorks application logging is under the control of the individual applications. For information on logging, see the related documentation for each application.

10.1.4 Configuration Agent Logs

The Configuration agent responsible for hosting the server's configuration issues logs to the */var/broadworks/logs/config* directory.

10.1.5 Other BroadWorks Logs

In general, all logs issued by BroadWorks components are located in the */var/broadworks/logs* directory.

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.

ACL	Access Control List
Admin	Administrator
API	Application Programming Interface
AS	Application Server
BW	BroadWorks
CAP	Client Application Protocol
CLI	Command Line Interface
EMS	Element Management System
FQDN	Fully Qualified Domain Name
HSS	Home Subscriber Server
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure Sockets
Hz	Hertz
IMS	IP Multimedia Subsystem
IP	Internet Protocol
NS	Network Server
NSOCI	Network Server Open Client Interface
NSOSS	Network Server Operations Support System
OCI	Open Client Interface
OCI-C	Open Client Interface-Call Control
OCI-P	Open Client Interface-Provisioning
OS	Operating System
PM	Performance Measurement
PS	Profile Server
SNMP	Simple Network Management Protocol
SSL	Secure Sockets Layer
URL	Uniform Resource Locator
Webapp	Web Application
WebDAV	Web-based Distributed Authoring and Versioning
XS	Execution Server
Xsi	Xtended Services Interface
Xsp	Xtended Services Platform

References

[1] BroadSoft, Inc. 2016. *BroadWorks Software Management Guide, Release 21.0*. Available from the BroadSoft Xchange at xchange.broadsoft.com.

[2] BroadSoft, Inc. 2016. *BroadWorks Element Management System Administration Guide, Release 21.0*. Available from the BroadSoft Xchange at xchange.broadsoft.com.

[3] BroadSoft, Inc. 2014. *BroadWorks Xtended Services Platform Configuration Guide, Release 21.0*. Available from the BroadSoft Xchange at xchange.broadsoft.com.

[4] BroadSoft, Inc. 2015. *BroadWorks XS Mode Configuration Guide, Release 21.0*. Available from the BroadSoft Xchange at xchange.broadsoft.com.

[5] BroadSoft, Inc. 2016. *BroadWorks Maintenance Guide*, *Release 21.0*. Available from the BroadSoft Xchange at xchange.broadsoft.com.

[6] BroadSoft, Inc. 2015. *BroadWorks External Portal Integration Guide*, *Release 21.0*. Available from the BroadSoft Xchange at xchange.broadsoft.com.

[7] BroadSoft, Inc. 2015. *BroadWorks Call Center Solution Guide*, *Release 21.0*. Available from the BroadSoft Xchange at xchange.broadsoft.com.

[8] BroadSoft, Inc. 2016. *BroadWorks Enhanced Call Logs Solution Guide*, *Release 21.0*. Available from the BroadSoft Xchange at xchange.broadsoft.com.

[9] BroadSoft, Inc. 2014. *BroadWorks Performance Measurements Interface Specification, Release 21.0*. Available from the BroadSoft Xchange at www.broadsoft.com/xchange.

[10] BroadSoft, Inc. 2015. *BroadWorks Fault and Alarm Interface Specification, Release 21.0*. Available from the BroadSoft Xchange at xchange.broadsoft.com.

[11] BroadSoft, Inc. 2010. *BroadWorks Name Service Integration Feature Description, Release 16.0*. Available from BroadSoft at xchange.broadsoft.com.

