Contents

1. Introduction
   Starting Network Gatekeeper ................................................................. 1-3
   Managing Network Gatekeeper ............................................................... 1-4
   Network Gatekeeper Administration Console ........................................... 1-4
      Starting the Network Gatekeeper administration console: ....................... 1-5
      Management objects page .......................................................................... 1-6
      Configuration and Provisioning ................................................................. 1-7
      WebLogic Network Gatekeeper Alarms page ............................................. 1-9
   WebLogic Network Gatekeeper and WebLogic SIP Server management console for
      SIP-based traffic paths .............................................................................. 1-11
      Starting the Network Gatekeeper administration console: ....................... 1-11
      Network Gatekeeper Integration Configuration page ................................. 1-11
   Patching Network Gatekeeper ................................................................. 1-12
      Getting the patch script ............................................................................ 1-13
      Running the patch tool .............................................................................. 1-13
      Re-deploying the ear file ............................................................................ 1-14
      Overall Configuration Workflow ............................................................... 1-14

2. Managing the Core Service and Administrative Users
   Overview of Network Gatekeeper Core and Network Gatekeeper Services ........ 2-2
   Network Gatekeeper Core .......................................................................... 2-2
   Network Gatekeeper services ...................................................................... 2-2
   Administrative Users .................................................................................. 2-3
Managing raised alarms .................................................. 5-36
Changing alarm severity levels for backwards compatible traffic paths .... 5-36
Managing external alarm listeners for backwards compatible traffic paths .. 5-37
Reference: attributes and operations for SLEE_alarm ......................... 5-37

6. Managing and Configuring Statistics and Transaction Licenses

About Statistics Generation and Reports ................................ 6-1
Overview of Statistics Reports ................................................ 6-2
System report to console ..................................................... 6-3
Report to file ................................................................. 6-3
Weekly system report ......................................................... 6-3
License limit log report ....................................................... 6-3
Operation, Maintenance, and Provisioning .................................. 6-5
Reference: attributes and operations for SLEE_statistics ................. 6-6
Transaction Types ........................................................... 6-15
Transaction Types for Enhanced Traffic Paths ............................. 6-17
Transaction Types for Backwards Compatible Traffic Paths ............ 6-19

7. Managing and Configuring the Global Counter Service

Configuration and Management ............................................ 7-1
Configuration workflow ..................................................... 7-1
Management operations ................................................... 7-1
Reference: attributes and operations for SLEE_global_counter .......... 7-2

8. Managing and Configuring the SCS Manager

Introduction ................................................................. 8-1
Configuration and Management ........................................ 8-1
Configuration workflow .................................................. 8-1
9. Managing and Configuring the SLEE Event Service

Configuration and Management ................................................. 9-1
  Configuration workflow .................................................. 9-1
  Management operations .................................................. 9-1
Reference: attributes and operations for SLEE_event.................... 9-2

10. Setting Up Geographic Redundancy

Introduction ................................................................. 10-1
Configuration Workflow .................................................... 10-1
Reference: attributes and operations for GeoRedundantService ....... 10-2

11. Setting up WS-Policy and JMX Policy

Introduction ................................................................. 11-1
Web Services Security ..................................................... 11-1
  Configuration workflow: WS-Policy .................................. 11-2
    Apply WS-Policy to a Web Service: Quick start ................. 11-3
    Remove WS-Policy from a Web Service ......................... 11-4
  Create and use custom a custom WS-Policy ....................... 11-6
    Available default WS-Policies ................................... 11-6
JMX Policy ...................................................................... 11-6
  Administrative Groups ................................................. 11-7
  Administrative Service Groups ................................. 11-7

12. Managing and Configuring the SNMP service

Introduction ................................................................. 12-1
Configuration and management ............................................ 12-3
13. Configuring WebLogic SIP Server Integration
   Introduction .................................................. 13-1
   Configuration Workflow .................................. 13-1
   Reference: attributes and operations .................. 13-2

14. Managing and Configuring the Trace Service
   Introduction to the Trace Service ....................... 14-1
   Basic tracing ................................................. 14-1
   Context trace ............................................... 14-2
   Reference: attributes and operations for Trace Service .... 14-3
   Operation: initialize ..................................... 14-8
   Log4J Hierarchies, loggers, and appenders .......... 14-10
   Configuring Trace for Access Tier servers .......... 14-11

15. Managing and Configuring Third Party Call Traffic Paths
   Third Party Call Traffic Paths ......................... 15-1
   The Enhanced Parlay X 2.1 Third Party Call traffic path with SIP integration .......... 15-1
      Configuration workflow .................................. 15-2
      Reference: attributes and operations for Plugin_third_party_call_sip ............. 15-2
   The Backwards Compatible Third Party Call Traffic Path with OSA MultiParty Call
      Control .................................................... 15-6
   Add the traffic path to Network tier ................. 15-7
   Configuration workflow .................................. 15-8
   Stateless adapter for the backwards compatible Third Party Call traffic path ........ 15-9
      Reference: attributes and operations for SESPA_callcontrol .............. 15-9
16. Managing and Configuring Call Notification Traffic Paths

Call Notification Traffic Paths. .................................................. 16-1
OSA MultiParty Call Control for Call Notification traffic path. .............. 16-1
  Configuration workflow ....................................................... 16-2
  Management and provisioning operations ............................... 16-2
  Reference: attributes and operations for Plugin_call_notification_parlay_mpcc16-2
SIP integration for the Parlay X 2.1 Call Notification traffic path. ........ 16-4
  Configuration workflow ....................................................... 16-4
  Reference: Attributes and operations for Plugin_call_notification_sip ...... 16-5

17. Managing and Configuring Short Messaging Traffic Paths

Short Messaging Traffic Paths. ................................................ 17-1
SMPP plug-in for Parlay X 2.1 Short Messaging. ............................ 17-1
  Configuration workflow ....................................................... 17-2
  Provisioning workflow ....................................................... 17-3
  Management operations ...................................................... 17-3
  Parameters that can be set using contextAttribute ..................... 17-3
  Reference: attributes and operations for Plugin_sms_SMPP .............. 17-5
OSA Multimedia Messaging plug-in for Parlay X 2.1 Short Messaging .... 17-25
  Configuration workflow ....................................................... 17-25
  Provisioning workflow ....................................................... 17-26
  Management operations ...................................................... 17-26
18. Managing and Configuring Multimedia Messaging Traffic Paths

Backwards Compatible Multimedia Messaging Traffic Paths ........................... 18-1

Mailboxes. ......................................................................................................... 18-1

Overall configuration workflow. ................................................................. 18-2

Stateless adapter for the backwards compatible Multimedia Messaging traffic path 18-3

Reference: attributes and operations for SESPA_px_messaging .............. 18-3

Reference: attributes and operations for SESPA_messaging .................... 18-5

Service capability for the backwards compatible Multimedia Messaging traffic paths . 18-6

Reference: attributes and operations for ESPA_messaging ....................... 18-6

OSA MultiMedia Messaging plug-in for the backwards compatible Multimedia

Messaging traffic path. .................................................................................. 18-16

Configuration workflow .................................................................................. 18-16

Reference: attributes and operations for Plugin_OSA_MM_MMS .......... 18-17

MM7 plug-in for the backwards compatible Multimedia Messaging traffic path. 18-22

Configuration workflow .................................................................................. 18-22

Reference: attributes and operations for Plugin_messaging_MM7 .......... 18-23

19. Managing and Configuring Payment Traffic Paths

Backwards compatible Payment Traffic Paths .............................................. 19-1

Overall configuration workflow ................................................................. 19-1

Service capability for backwards compatible Payment traffic path .......... 19-2

Reference: attributes and operations for ESPA_charging ....................... 19-2

Configuration workflow .................................................................................. 19-3

Reference: attributes and operations for Plugin_OSA_Charging .............. 19-4
20. Managing and Configuring Terminal Status Traffic Paths

- Terminal Status Traffic Paths ................................................... 20-1
  OSA User Status plug-in for the Parlay X 2.1 Terminal Status traffic path ........... 20-1
    Configuration workflow ....................................................... 20-2
    Reference: attributes and operations for Plugin_terminal_status_parlay_us . 20-2

21. Managing and Configuring Terminal Location Traffic Paths

- Backwards Compatible Terminal Location Traffic Paths .......................... 21-1
  Overall configuration workflow .............................................. 21-2
  Stateless adapter for backwards compatible Terminal Location traffic paths .... 21-2
    Reference: attributes and operations for SESPA_userlocation ................. 21-3
  Service capability for backwards compatible Terminal Location traffic paths ... 21-3
    Reference: attributes and operations for ESPA_userlocation ................. 21-3
  MLP plug-in for backwards compatible Parlay X 2.1 Terminal Location traffic path . 21-7
    Mode of operation .......................................................... 21-7
    Request buffering ......................................................... 21-8
    Configuration workflow ................................................... 21-8
    Reference: attributes and operations for Plugin_user_location_MLP .......... 21-9
  OSA User Location plug-in for backwards compatible Terminal Location traffic path . 21-18
    Configuration workflow ................................................... 21-18
    Reference: attributes and operations for Plugin_OSA_UL .................... 21-19

22. Managing and Configuring Call Handling Traffic Paths

- Call Handling Traffic Path .................................................... 22-1
  OSA MultiParty Call Control and OSA Generic User Interaction plug-in for Call
    Handling ................................................................. 22-1
23. Managing and Configuring Audio Call Traffic Path
   Audio Call Traffic Path ............................................................ 23-1
   OSA MultiParty Call Control and OSA Generic User Interaction plug-in for Audio Call
   23-1
   Configuration workflow ......................................................... 23-1
   Reference: attributes and operations for Plugin_audio_call_parlay_mpcc_cui 23-3

24. Managing and Configuring the Presence Traffic Path
   Presence Traffic Path .............................................................. 24-1
   SIP integration for the Parlay X 2.1 Presence traffic path ................. 24-1
   URI cache .............................................................................. 24-2
   Subscriptions cache ............................................................... 24-2
   Notifications cache ............................................................... 24-2
   Configuration workflow ......................................................... 24-2
   Provisioning workflow ......................................................... 24-3
   Management operations ....................................................... 24-3
   Attributes and operations for Plugin_presence_sip .................... 24-3

25. Managing and Configuring WAP Push Traffic Paths
   Backwards compatible WAP Push traffic paths .......................... 25-1
   Overall configuration workflow ............................................... 25-1
   Stateless adapter for backwards compatible WAP Push traffic path .... 25-2
   Reference: attributes and operations for SESPA_message_sender ...... 25-2
   Service capability for backwards compatible WAP Push traffic path ... 25-3
   Attributes and operations for ESPA_message_sender ................. 25-3
   PAP plug-in for WAP Push backwards compatible traffic paths ....... 25-4
Configuration workflow ................................. 25-5
Reference: attributes and operations for Plugin_message_sender_PAP .... 25-5
OSA/Parlay Multimedia Messaging plug-in for WAP Push backwards compatible traffic paths .................................................. 25-7
Configuration workflow ................................. 25-7
Reference: attributes and operations for Plugin_OSA_MM_WAP ........ 25-8

26. Managing and Configuring routes and Node SLAs

Introduction .............................................. 26-1
Execution and evaluation flow ......................... 26-2
How address ranges are specified ...................... 26-3
Configuration workflow ................................. 26-4
Administration of plug-in routes ....................... 26-4
Updating and loading a global node SLA ............... 26-5
Configuring the plug-in manager ....................... 26-5
Reference: attributes and operations for Plugin_manager ............... 26-5
Writing Node SLAs ........................................ 26-12
SLA file overview ....................................... 26-12
Service provider node SLA ............................. 26-12
Node contract data ...................................... 26-14
Node SLA file ............................................ 26-14
Global contract data ..................................... 26-15
Node SLA tags ............................................ 26-16

<startDate> ....................................... 26-16
<endDate> ........................................ 26-16
<nodeID> ............................................. 26-16
<nodeRestrictions> .................................... 26-16
<nodeRestriction> ..................................... 26-17
27. Managing and Configuring OSA/Parlay Gateway Connections

Understanding OSA/Parlay Gateway and account mappings ........................................... 27-1
Connection model .................................................................................................................. 27-1
Information and certificate exchange with OSA/Parlay administrator ............................... 27-3
Overall workflow when connecting to an OSA Gateway ..................................................... 27-4
Adding an OSA Gateway ....................................................................................................... 27-5
Adding an OSA Gateway Connection .................................................................................... 27-5
Creating an OSA client ......................................................................................................... 27-6
Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS ............................. 27-6
Reference: attributes and operations for Plugin_OSA_Access ............................................ 27-6

28. Callable Policy Web Service

Introduction ............................................................................................................................. 28-2
Callable Policy Web Service interface definition ............................................................... 28-3
Endpoints ............................................................................................................................. 28-3
Detailed service description ................................................................................................. 28-3
Policy Evaluation .................................................................................................................. 28-3
Policy management .............................................................................................................. 28-4
XML Schema data type definition ....................................................................................... 28-4
AdditionalDataType structure ............................................................................................ 28-4
AdditionalDataType enumeration ....................................................................................... 28-4
Interface: Policy ................................................................................................................... 28-5
Operation: evaluate ............................................................................................................. 28-5
Interface: PolicyManagement ................................................... 28-7
Operation: reloadXmlDriver ................................................... 28-7
Operation: viewRuleFile ...................................................... 28-7
Operation: deleteRuleFile .................................................... 28-8
Operation: loadRules .......................................................... 28-9
Operation: listRuleFiles ...................................................... 28-10
Rule files .............................................................................. 28-11

29.Managing the Policy Service
Introduction ........................................................................ 29-1
Configuration workflow ....................................................... 29-1
Reference: attributes and operations for Policy ....................... 29-2

30.Writing OAM Scripts and using JMX
Working in interactive mode .................................................. 30-2
Starting WLST and connecting to Network Gatekeeper .............. 30-2
Exiting WLST ...................................................................... 30-2
Changing an attribute .......................................................... 30-2
Invoking an operation ......................................................... 30-3
Scripting WLST .................................................................. 30-4
Using JMX for Management ............................................... 30-5

31.Creating an EDR listener
Overview of External EDR listeners ....................................... 31-1
Example using a pure JMS listener ......................................... 31-3
Example using JMSListener utility with no filter ..................... 31-3
Using JMSListener utility with a filter .................................... 31-4
Description of EDR listener utility ........................................ 31-5
Class JMSListener .............................................................. 31-6
32. Rolling Upgrade (MP01)

Prepare the domain ................................................................. 32-1
Create the clusters that will hold the updated version .................. 32-2
Edit the config.xml file ......................................................... 32-2
Define migration settings ..................................................... 32-3
Deploy the new version ......................................................... 32-4
Modify the file APP-INF/classes/wlng.properties ...................... 32-4
Move initial access tier and network tier server to the new cluster . 32-4
Install the updates in the new clusters .................................... 32-4
Move the servers to the new cluster ....................................... 32-5
Cleanup the old applications ................................................. 32-6
Limitations & known issues .................................................. 32-6
Introduction

This guide describes Network Gatekeeper from operational, management and maintenance aspects.

This chapter describes how to start WebLogic Network Gatekeeper and how to use the Network Gatekeeper Management Console:

- Starting Network Gatekeeper
- Managing Network Gatekeeper
  - Network Gatekeeper Administration Console
- Patching Network Gatekeeper
- Overall Configuration Workflow

The following chapters describe the configuration and management of the Network Gatekeeper Core and Core utility services:

- Managing the Core Service and Administrative Users
- Managing the States of Services
- Managing and Configuring Budgets
- Managing and Configuring EDRs, CDRs and Alarms
- Managing and Configuring Statistics and Transaction Licenses
- Setting Up Geographic Redundancy
Introduction

- Managing and Configuring the SLEE Event Service
- Managing and Configuring the Global Counter Service
- Managing and Configuring the SCS Manager
- Managing and Configuring the SNMP service
- Managing and Configuring the Trace Service

The following chapters describe the configuration and management of traffic paths:

- Managing and Configuring Third Party Call Traffic Paths
- Managing and Configuring Call Notification Traffic Paths
- Managing and Configuring Short Messaging Traffic Paths
- Managing and Configuring Multimedia Messaging Traffic Paths
- Managing and Configuring Payment Traffic Paths
- Managing and Configuring Terminal Status Traffic Paths
- Managing and Configuring Terminal Location Traffic Paths
- Managing and Configuring Call Handling Traffic Paths
- Managing and Configuring Audio Call Traffic Path
- Managing and Configuring the Presence Traffic Path
- Managing and Configuring WAP Push Traffic Paths
- Managing and Configuring routes and Node SLAs
- Managing and Configuring OSA/Parlay Gateway Connections
- Configuring WebLogic SIP Server Integration
- Installing and using multiple instances of a plug-in
- Setting up WS-Policy and JMX Policy

The following chapters describe controlling and extending policy:

- Callable Policy Web Service
• The following chapter gives an introduction on how to write scripts for managing and configuring Network Gatekeeper: Writing OAM Scripts and using JMX

• The following chapter describes how to create an external EDR listener: Creating an EDR listener.

Starting Network Gatekeeper

Because a typical WebLogic Network Gatekeeper domain contains multiple Access and Network tier servers, with dependencies among the different server types, you should generally follow this sequence when starting up a domain:

1. Start the Administration Server for the domain.

   Start the Administration Server in order to provide the initial configuration to Access and Network tier servers in the domain. The Administration Server can also be used to monitor the startup/shutdown status of each Managed Server. You generally start the Administration Server by using either the startAdminServer script installed with the Configuration Wizard, or a custom startup script.

2. Start Network tier servers in each partition.

   The Access tier cannot function until servers in the Network tier are available.


   WARNING: All servers should be started and available before opening the system to production network traffic.

   You generally start Network and Access tier servers by using the startManagedWebLogic script installed with the Configuration Wizard, a custom startup script, or the Management Console in conjunction with an instance of Node Manager running on each machine.

   To use the node manager to start Network Gatekeeper servers, start the Node Manager on the computer that will host the Managed Server you want to start. Go to the domain's Administration Console. Under Environment, select the managed servers you want to start.

   To use the startManagedWebLogic script, you must specify the name of the server to startup, as well as the URL of the Administration Server for the domain, as in:

   startManagedWebLogic.sh networknode0-0 t3://adminhost:7001

   For more information about how to start and stop servers see Starting and Stopping Servers: Quick Reference and Starting and Stopping Servers.
Resolving startup errors

**Note:** If this error occurs during startup (in a stack trace or elsewhere):

```java
org.omg.CORBA.COMM_FAILURE: bind() failed: java.net.BindException: Address already in use: JVM_Bind
 minor code: 0x4f4f0008 completed: No
```

The CORBA port is probably in conflict with another process. Make sure no other processes are using the default CORBA port of Network Gatekeeper. If this is not possible, the CORBA port can be changed using the following steps:

1. Edit the setDomainEnv.cmd/sh file in $DOMAIN_HOME/bin
2. Locate the definition:
   ```
   @REM set JAVA_OPTIONS=%JAVA_OPTIONS% -Dwlng.corba.mgmt_port=10007
   ```
3. Un-comment this line and change the port number.

Managing Network Gatekeeper

Network Gatekeeper is built using WebLogic Server. All management functionality provided by WebLogic server is available, and, in addition, there is a management console section specifically for managing the Network Gatekeeper aspects of your installation: see Network Gatekeeper Administration Console.

There is also a management console for managing the Network Gatekeeper parts that execute in the WebLogic SIP server as a part of the WebLogic Network Gatekeeper and WebLogic SIP Server integration: see WebLogic Network Gatekeeper and WebLogic SIP Server management console for SIP-based traffic paths.

Network Gatekeeper Administration Console

The Network Gatekeeper Administration Console provides a graphical user interface for modifying Network Gatekeeper. The Administration Console for Network Gatekeeper is similar to the core console available in WebLogic Server.

At least one Network tier server must be started prior to logging in to WebLogic Server Administration Console. Network Gatekeeper servers started after logging in to the WebLogic Server Administration Console are not displayed in Network Gatekeeper console. To see them, you must login again.
Starting the Network Gatekeeper administration console:

Use a supported web browser to go to http://<server>:<port>/console where <server> is the instance you have set up as your Administration Server.

Login using your log-in credentials.

Note: If it is the first time Network Gatekeeper is started, use the username weblogic and password weblogic as defined when configuring the domain. Then create an administrative user using the instructions given in Managing the Core Service and Administrative Users, section Managing Network Gatekeeper Administrative Users and remove the user weblogic. Either remove or change password for the WebLogic Server user: see http://edocs.bea.com/wls/docs92/secwlres/secroles.html.

All Network Gatekeeper configuration and monitoring is provided via the following nodes in the left pane of the console, in the Domain Structure group:

- WLNG — container for all Network Gatekeeper servers
- <Server Name> — one entry per Network Gatekeeper server.
- <Server Name> -> Services — Clicking on this link displays the Management objects page which displays all OAM objects available for the selected Network Gatekeeper server. Some configuration settings are cluster-wide while other settings are per server.
- <Server Name> -> Log4J — Clicking on this link displays the Log4J configuration page.
- Alarms — clicking this link displays the WebLogic Network Gatekeeper Alarms page.
Management objects page

The Management Object page contains a list of all management objects applicable for the selected Network Gatekeeper server.

By clicking on a management object, the corresponding Configuration and Provisioning page for the management object is displayed: see Configuration and Provisioning.
**Configuration and Provisioning**

The *Configuration and Provisioning* page displays a list of *Attributes*, either read-only or read-write, and a set of operations.

**Note:** In this document, read-only attributes are indicated by *(r)* after the name.

Read-write attributes have a checkbox next to them.

To change an attribute:

1. Check the check box.
2. Enter the new value in the entry-field.
3. Click **Update Attributes**.

**Figure 1-3** Configuration and provisioning page for a managed object

Operations available for the managed object are found in the dropdown-list labeled **Select An Operation**.

Operations either display data, set data or perform an actual task.

To perform an operation:

1. Select the operation from the dropdown-list **Select An Operation**.
   
The entry-fields for the operation are displayed.

2. Enter the information in the entry fields.

3. Click **Invoke** to perform the operation.
WebLogic Network Gatekeeper Alarms page

The WebLogic Network Gatekeeper Alarms page displays alarms emitted from Network Gatekeeper.

It is possible to filter the output to the page on a set of criteria:

- Severity of alarm.
- The server from which the alarm originates.

From the dropdown-list labeled **Severity**, choose which severity level to display:

- **ALL**
- **WARNING**
- **MINOR**
- **MAJOR**
- **CRITICAL**

From the dropdown-list labeled **Server**, choose the server from which to display alarms: the options are **ALL**, for all servers or you can select an individual server.
Figure 1-5 Alarms page.

Weblogic Network Gatekeeper Alarms

Severity [ALL]  Server [ALL]

2007-03-01 17:03:53.335 Alarm On wlng2

-----> Alarm emitted by WING
    Alarm ID: 255016328899854431
    Severity: WARNING
    Identifier: 25707
    Info: Receiver connection successfully established
    Time: 2 Mar 2007 01:03:33 GMT

2007-03-01 17:03:53.333 Alarm On wlng2

-----> Alarm emitted by WING
    Alarm ID: 255016328899854430
    Severity: WARNING
    Identifier: 25710
    Info: Transmitter connection successfully established
    Time: 2 Mar 2007 01:03:33 GMT

2007-03-01 17:03:42.746 Alarm On wlng1

-----> Alarm emitted by WING
    Alarm ID: 326792447961071711
    Severity: WARNING
    Identifier: 25707
    Info: Receiver connection successfully established
    Time: 2 Mar 2007 01:03:25 GMT
**WebLogic Network Gatekeeper and WebLogic SIP Server management console for SIP-based traffic paths**

Network Gatekeeper also has an Administration Console for Network Gatekeeper related configuration in WebLogic SIP Server. This is only relevant for the SIP Server parts of Network Gatekeeper traffic paths, and handles settings for the modules deployed in WebLogic SIP server.

**Starting the Network Gatekeeper administration console:**

Use a Web browser to go to `http://<server>:<port>/console` on the WebLogic SIP Server Administration server.

Login using your log-in credentials.

The Administration Console is similar to the core console available in WebLogic Server. All Network Gatekeeper specific configuration is done via the **WLNG Integration** node in the left pane of the console, in the **Domain Structure** group.

![Figure 1-6 Domain Structure -links to management console for Network Gatekeeper SIP Server integration modules deployed in WebLogic SIP Server](image)

**Network Gatekeeper Integration Configuration page**

The Network Gatekeeper Integration Configuration page contains a set of tabs that contain general integration configuration attributes and traffic path specific configuration attributes.

These attributes use the Lock and Edit functionality provided by WebLogic Server Administration console.
### Patching Network Gatekeeper

From time to time, patches are issued for aspects of the two core Network Gatekeeper application files, `wlng_at.ear` and `wlng_nt.ear`. Installing these patches is accomplished using a script, `patch.sh`, which automatically opens the ear, makes the necessary adjustments, and re-packages the ear. Once the patch has been applied, you simply re-deploy the ears.

1. **Getting the patch script**
2. **Running the patch tool**
3. **Re-deploying the ear file**

Below are some recommendations on how to manage patches:

- After having installed Network Gatekeeper you should make a copy of the `wlng_nt.ear` and `wlng_at.ear`. 

---

#### Figure 1-7  Network Gatekeeper Integration Configuration page in WebLogic SIP server

**WLNG Integration Configuration**

<table>
<thead>
<tr>
<th>General</th>
<th>Third Party Call Control</th>
<th>Call Notification</th>
<th>Presence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WLNG JNDI URL:</th>
<th>localhost:7002</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Username:</th>
<th>weblogic</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Password:</th>
<th>weblogic</th>
</tr>
</thead>
</table>

---
When you receive a new patch, reapply all patches you have received, including the new patch, to wlng_nt.ear or wlng_at.ear.

When receiving a combo patch you should remove the atomic patches included in the combo patch from the list of patches that should be applied.

### Getting the patch script

Patching is done using `patch.sh`, a script that can be found in `<wlng_home>/common/bin/`. In the default installation, this would be `wlng300/common/bin/patch.sh`.

### Running the patch tool

Specific patch jar files, conventionally named `CRRxxx.jar` where CRRxxx is the change request identifier associated with the issue that produced the update, are acquired through BEA Support. The user running the script must have read/write permission for the jar file and the ear file that is being patched (located in `<domain_home>/`), and write permission for the directory in which the tool is being run.

The script should be executed in a bash shell on the Administration Server. The syntax is as follows:

- To see the version of the patch file itself:
  ```bash
  patch.sh -version <patch_path>
  ```
- To see the version of the target ear file:
  ```bash
  patch.sh -version <ear_path>
  ```
- To see the ear version that is currently running:
  ```bash
  patch.sh -version <ear_path> <server_name>
  ```
- To build the patch file into to the target ear file (that is, to patch the ear file):
  ```bash
  patch.sh -build <ear_path> <patch_path>
  ```
- To get usage help:
  ```bash
  patch.sh -help
  ```

The full path of patch file, for example `<bea_home>/wlng300/_patch/CRRxxx.jar`
<ear_path>
The full path of the ear file being patched, for example
<bea_home>/user_projects/domains/wlngdomain/wlng_nt.ear
</ear_path>

<server_name>
The name of the server running the ear file you wish to query, for example WLNG_NT1
</server_name>

The previous version of the ear file is saved as ear.back, in its original directory.

Re-deploying the ear file

Once the patch tool has complete running, you must re-deploy the ear file for the changes to take
effect. To redeploy using the Management Console:

1. In the Management Console, click on Deployments in the Domain Structure group in the
left navigation bar.
2. Select the ear file you wish to re-deploy, either wlng_nt.ear or wlng_at.ear.
3. From the drop down Stop menu, select Force Stop.
4. Restart the newly patched ear by selecting Update.

Overall Configuration Workflow

This section describes an overall workflow for configuring Network Gatekeeper.

1. Setup administrative users - see Managing the Core Service and Administrative Users, section
Managing Network Gatekeeper Administrative Users.
2. Configure Network Gatekeeper core service - see Managing the Core Service and
Administrative Users, section Network Gatekeeper Core.
3. Configure Network Gatekeeper Core utility services - see:
   - Managing and Configuring Budgets
   - Managing and Configuring EDRs, CDRs and Alarms
   - Managing and Configuring Statistics and Transaction Licenses
   - Managing and Configuring the SLEE Event Service
   - Managing and Configuring the Global Counter Service
   - Managing and Configuring Statistics and Transaction Licenses
4. Configure Network Gatekeeper traffic paths. **Note:** you only need to configure the paths that are used in your deployment.
   - Managing and Configuring Third Party Call Traffic Paths
   - Managing and Configuring Call Notification Traffic Paths
   - Managing and Configuring Short Messaging Traffic Paths
   - Managing and Configuring Multimedia Messaging Traffic Paths
   - Managing and Configuring Payment Traffic Paths
   - Managing and Configuring Terminal Status Traffic Paths
   - Managing and Configuring Terminal Location Traffic Paths
   - Managing and Configuring Call Handling Traffic Paths
   - Managing and Configuring Audio Call Traffic Path
   - Managing and Configuring the Presence Traffic Path
   - Managing and Configuring WAP Push Traffic Paths
   - Managing and Configuring routes and Node SLAs
   - Managing and Configuring the SCS Manager
   - Managing and Configuring OSA/Parlay Gateway Connections

5. Configure Security for Web Services and OAM MBeans - see Setting up WS-Policy and JMX Policy for an introduction and pointers to relevant information.

6. If you are setting up geographically redundant site pairs, use the instructions in Setting Up Geographic Redundancy.

Once the system is configured, move on to provisioning of service providers and applications, as described in Managing Service Providers and Applications.
CHAPTER 2

Managing the Core Service and Administrative Users

The following section describes how to manage the Network Gatekeeper Core and how to set up administrative users for Network Gatekeeper.

- Overview of Network Gatekeeper Core and Network Gatekeeper Services
  - Network Gatekeeper Core
  - Network Gatekeeper services
  - Administrative Users
- Managing Core (SLEE) State
- Viewing Memory and Disk Space Utilization
- Viewing and Managing SLEE Load and Resource Utilization
- Managing SLEE Services
- Managing Network Gatekeeper Administrative Users
- Reference: attributes and operations for SLEE
Overview of Network Gatekeeper Core and Network Gatekeeper Services

Network Gatekeeper Core

Network Gatekeeper is built with a highly modular software architecture. Modules within the Network Gatekeeper run as J2EE modules, but a certain number of them also have some additional characteristics which qualify them as Network Gatekeeper services, or SLEE services. Network Gatekeeper itself manages common functions for these SLEE services: lifecycle, service supervision, and database supervision.

Network Gatekeeper Core has one of two states: RUNNING, or SUSPENDED. When a SLEE process is started up, it is in the RUNNING state.

The SUSPENDED state temporarily stops all requests into and out of Network Gatekeeper Core without actually stopping the started or activated Network Gatekeeper services.

Network Gatekeeper services

Some software modules originally deployed in the 2.2 release are, for 3.0, based on the “backwards compatibility” model. These modules are installed and run in Network Gatekeeper Core as Network Gatekeeper (formerly SLEE) services. A Network Gatekeeper service can have one of five states:

- Installed
  The service software is installed in Network Gatekeeper.
- Started
  The service is started and available in the Console but cannot send and receive requests.
- Activated
  The service is running and can send and receive requests.
- Suspended
  The service is activated but cannot receive new service requests.
- Error
The service has raised too many critical alarms and has been taken out of service by Network Gatekeeper Core. The allowed number of critical alarms is configured at service installation.

In case of a Network Gatekeeper Core restart, the service’s restart order and previous operating states are retrieved from the database.

**Administrative Users**

Management of Network Gatekeeper and Network Gatekeeper services is performed by administrative users. Each registered user is assigned a user level which controls the type of OAM methods to which the user has access in Network Gatekeeper management console. There are four defined user levels:

- Anonymous
- Read only
- Standard read and write
- Administrator

Administrative users are also assigned to service groups. Each service group consists of a number of related SLEE services.

A user can be connected to more than one service group and a SLEE service can be a member of more than one service group.

**Managing Core (SLEE) State**

To view a Network Gatekeeper server (SLEE) name, use `Attribute: Name (r)`. To view Network Gatekeeper server (SLEE) state, use `Attribute: SLEEState (r)`.

To change a Network Gatekeeper server (SLEE) state, use:

- **Operation**: `resumeAccess`
- **Operation**: `suspendAccess`

**Viewing Memory and Disk Space Utilization**

To view memory information, use:

- **Attribute**: `FreeMemory (r)`.
Managing the Core Service and Administrative Users

- **Attribute:** TotalMemory (r).
- To view disk space utilization, use **Operation:** getFreeDiskSpace.

Viewing and Managing SLEE Load and Resource Utilization

The following information is available:

- Current load
- Average load for a specified time period
- Load history for a specified time period, presented as a list of 5 minute averages

  **Note:** These load values are presented as a percentage of the maximum load.

To view current load, use **Attribute:** Load (r).
To view average load, use **Operation:** getAverageLoad.
To view load history, use **Operation:** listLoadAverages.
To remove load history data from the database, use **Operation:** deleteLoadAverages.

Managing SLEE Services

To view the Network Gatekeeper services, see **Operation:** getServices.
To view the state of a Network Gatekeeper service, see **Operation:** getServiceState.
To view the version of a Network Gatekeeper service, see **Operation:** getServiceVersion.
To change the state of a service, see Managing the States of Services.

Managing Network Gatekeeper Administrative Users

To create an administrative service group, use **Operation:** createServiceGroup.
To add a service to an administrative service group, use **Operation:** addUserToGroup.
To create an administrative user, use **Operation:** addUser.
To add an administrative user to an administrative service group, use **Operation:** addUserToGroup.
To list administrative users, use **Operation:** listUsers.
To change password for an administrative user, use Operation: changeUserPassword.
To list the service groups to which an administrative user has access, Operation: listGroupsForUser.
To list administrative services in an administrative service group, use Operation: listServicesInGroup.
To remove an administrative user from an administrative service group, use Operation: removeUserFromGroup.
To list administrative service groups for the administrative user, use Operation: listGroupsForUser.
To remove an administrative user from an administrative service group, use Operation: removeUserFromGroup.
To delete an administrative user, use Operation: deleteUser.
To list administrative users in an administrative service group, use Operation: listUserInGroup.
To view all administrative services in an administrative service group, use Operation: listServicesInGroup.
To remove an administrative service from an administrative service group, Operation: removeServiceFromGroup.
To delete an administrative service group, use Operation: deleteServiceGroup

Reference: attributes and operations for SLEE

Below is a list of attributes and operations for configuration and maintenance.

- Attribute: SLEEUptime (r)
- Attribute: Load (r)
- Attribute: ZombieSupervisorAssumeDeadTime
- Attribute: ZombieSupervisorHeartbeatTime
- Attribute: FreeMemory (r)
- Attribute: Name (r)
- Attribute: TotalMemory (r)
Managing the Core Service and Administrative Users

- Attribute: SLEEState (r)
- Attribute: SLEEPProcessUptime (r)
- Operation: addServiceToGroup
- Operation: addUser
- Operation: addUserToGroup
- Operation: changeUserPassword
- Operation: createServiceGroup
- Operation: deleteLoadAverages
- Operation: deleteServiceGroup
- Operation: deleteUser
- Operation: doGarbageCollect
- Operation: getAverageLoad
- Operation: getFreeDiskSpace
- Operation: getServiceState
- Operation: getServiceVersion
- Operation: getServices
- Operation: listGroupsForUser
- Operation: listLoadAverages
- Operation: listServiceGroups
- Operation: listServicesInGroup
- Operation: listUserInGroup
- Operation: listUsers
- Operation: removeServiceFromGroup
- Operation: removeUserFromGroup
- Operation: resumeAccess
Operation: suspendAccess

Attribute: SLEEUptime (r)
Scope: Server
Unit: millisecond.
Displays the time elapsed since Network Gatekeeper Core was started.

Attribute: Load (r)
Scope: Server
Unit: percent
Displays the load level.

Attribute: ZombieSupervisorAssumeDeadTime
Scope: Cluster
Format: int
Unit: seconds
Specifies the default time out for the zombie object supervisor. After this period elapses, zombie objects are declared dead.

Attribute: ZombieSupervisorHeartbeatTime
Scope: Cluster
Format: int
Unit: Seconds
Specifies the time between object supervision heartbeats. The heartbeats are sent to non-responding objects (zombies).

Attribute: FreeMemory (r)
Scope: Server
Format: int
Unit: bytes
Displays the amount of free memory.

**Attribute: Name (r)**
Scope: Server
Displays the server name. This name is defined at install time. This name is appended to service names to create a unique service instance name when the service is installed in multiple servers.

**Attribute: TotalMemory (r)**
Scope: Server
Unit: bytes
Displays the total amount of memory in the Java Virtual Machine.

**Attribute: SLEEState (r)**
Scope: Server
Display the current state of Network Gatekeeper Core. See [Network Gatekeeper Core](#).

**Attribute: SLEEPProcessUptime (r)**
Scope: Cluster
Format: int
Unit: milliseconds
Displays elapsed time since the Network Gatekeeper process started.

**Operation: addServiceToGroup**
Scope: Cluster
Adds a Network Gatekeeper service to an administrative group. See [Administrative Users](#)
Signature:
```
addServiceToGroup(service: String, group: String)
```
Table 2-1  addServiceToGroup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>Name of service.</td>
</tr>
<tr>
<td>group</td>
<td>Name of group.</td>
</tr>
</tbody>
</table>

**Operation: addUser**

Scope: Cluster

Adds a Network Gatekeeper administrative user. See Administrative Users.

Signature:

```java
addUser(newUsername:String, newPasswordKey: String, confirmPasswordKey: String, userLevel: int)
```

Table 2-2  addUser

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>newUsername</td>
<td>User name.</td>
</tr>
<tr>
<td>newPasswordKey</td>
<td>Password</td>
</tr>
<tr>
<td>confirmPasswordKey</td>
<td>Password (for confirmation)</td>
</tr>
</tbody>
</table>
| userLevel    | Defines the user level when administrating Network Gatekeeper Core, Core utilities and traffic paths. Use:  
- 0 for Unauthorized user  
- 333 for Read only user  
- 666 for Standard read/write user  
- 100 for Administrator user |
**Operation: addUserToGroup**

Scope: Cluster

Adds a Network Gatekeeper administrative user to an administrative group. See Administrative Users.

Signature:

\[ \text{addUserToGroup}(\text{group : String}, \text{user: String}) \]

**Table 2-3  addUser**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>Name of group.</td>
</tr>
<tr>
<td>user</td>
<td>User ID.</td>
</tr>
</tbody>
</table>

**Operation: changeUserPassword**

Scope: Cluster

Changes the password for an existing Network Gatekeeper administrative user. See Administrative Users.

Signature:

\[ \text{changeUserPassword}(\text{group : String}, \text{user: String}) \]

**Table 2-4  changeUserPassword**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userName</td>
<td>User ID</td>
</tr>
<tr>
<td>oldPasswordKey</td>
<td>Current password.</td>
</tr>
</tbody>
</table>
Reference: attributes and operations for SLEE

### Operation: createServiceGroup

**Scope:** Cluster

Creates a Network Gatekeeper administrative service group.

**Signature:**

```java
createServiceGroup(group :String, description: String)
```

### Table 2-4 changeUserPassword

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>newPasswordKey</td>
<td>New password.</td>
</tr>
<tr>
<td>confirmPasswordKey</td>
<td>New password (for confirmation)</td>
</tr>
</tbody>
</table>

### Operation: deleteLoadAverages

**Scope:** Server

Deletes information about load averages from database.

**Signature:**

```java
deleteLoadAverages(endTime :String)
```
Managing the Core Service and Administrative Users

Table 2-6 deleteLoadAverages

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>endTime</td>
<td>Remove entries up to this time.</td>
</tr>
</tbody>
</table>

**Operation: deleteServiceGroup**

Scope: Cluster

Deletes a Network Gatekeeper administrative service group.

Signature:

```
deleteServiceGroup(group :String)
```

Table 2-7 deleteServiceGroup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>Administrative group ID.</td>
</tr>
</tbody>
</table>

**Operation: deleteUser**

Scope: Cluster

Deletes a Network Gatekeeper administrative user. See Administrative Users.

Signature:

```
deleteUser(userToDelete :String)
```
Reference: attributes and operations for SLEE

**Table 2-8 deleteUser**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userToDelete</td>
<td>Administrative user ID.</td>
</tr>
</tbody>
</table>

**Operation: doGarbageCollect**

Scope: Server

Performs JVM garbage collection.

Signature:

doGarbageCollect()

**Table 2-9 doGarbageCollect**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: exitAndTerminateJVM**

Scope: Server

Deprecated. Do not use.

Exits the JVM, given that the SLEE is in state SHUTDOWN.

Signature:

exitAndTerminateJVM()
Operation: getAverageLoad

Scope: Server

Displays the average load on the server during a given time interval.

Signature:

getAverageLoad(startTime: String, endTime: String)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTime</td>
<td>Start time.</td>
</tr>
<tr>
<td>endTime</td>
<td>End time.</td>
</tr>
</tbody>
</table>

If empty string, startTime plus one hour is used.

Operation: getFreeDiskSpace

Scope: Server

Unit: kiloByte

Displays the free disk space for the disk partition on the given path.

Signature:

getFreeDiskSpace(path: String)
Table 2-12  getFreeDiskSpace

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Path to a directory in a partition or partition root path.</td>
</tr>
</tbody>
</table>

**Operation: getResourceShareOrbHost**

Scope: Server

Deprecated.

Displays the resource sharing context ORB host setting.

Signature:

```java
getResourceShareOrbHost(shareName: String)
```

Table 2-13  getResourceShareOrbHost

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shareName</td>
<td>ID of a resource share.</td>
</tr>
</tbody>
</table>

**Operation: getResourceShareOrbPoolSize**

Scope: Server

Deprecated.

Displays the resource sharing context ORB pool size setting.

Signature:

```java
getResourceShareOrbPoolSize(shareName: String)
```
Managing the Core Service and Administrative Users

System Administrator’s Guide

Table 2-14 getResourceShareOrbPoolSize

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shareName</td>
<td>ID of a resource share.</td>
</tr>
</tbody>
</table>

**Operation: getResourceShareOrbPort**

Scope: Server

Deprecated.

Displays the resource sharing context ORB port setting.

Signature:

getresourceShareOrbPort(shareName: String)

Table 2-15 getResourceShareOrbPort

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shareName</td>
<td>ID of a resource share.</td>
</tr>
</tbody>
</table>

**Operation: getServiceState**

Scope: Server

Displays the state of a given Network Gatekeeper service. See Network Gatekeeper services for a description of service states.

Signature:

getServiceState(serviceName : String)
Operation: `getServiceVersion`

Scope: Server

Displays the version attribute for a service. This attribute is defined in the deployment descriptor for the specified service.

Signature:

```
getServiceVersion(serviceName : String)
```

### Table 2-16 `getServiceState`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>The name of the module.</td>
</tr>
</tbody>
</table>

### Operation: `getServices`

Scope: Server

Displays the names of service in a given state. See Network Gatekeeper services for information about states.

Signature:

```
getServices(serviceState : String)
```
Managing the Core Service and Administrative Users

**Table 2-18 getServices**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceState</td>
<td>The state. Use:</td>
</tr>
<tr>
<td></td>
<td>State should be one of:</td>
</tr>
<tr>
<td></td>
<td>• -1 for SERVICE_UNINSTALLED</td>
</tr>
<tr>
<td></td>
<td>• 0 for SERVICE_INSTALLED</td>
</tr>
<tr>
<td></td>
<td>• 1 for SERVICE_STARTED</td>
</tr>
<tr>
<td></td>
<td>• 2 for SERVICE_ACTIVATED</td>
</tr>
<tr>
<td></td>
<td>• 3 for SERVICE_UNKNOWN</td>
</tr>
<tr>
<td></td>
<td>• 4 for SERVICE_SUSPENDED</td>
</tr>
<tr>
<td></td>
<td>• 5 for SERVICE_ERROR</td>
</tr>
</tbody>
</table>

**Operation: listGroupsForUser**

Scope: Cluster

Displays the names of all groups a specific Network Gatekeeper administrative user belongs to.

See Administrative Users.

Signature:

```java
listGroupsForUser(user: String)
```

**Table 2-19 listGroupsForUser**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>User ID</td>
</tr>
</tbody>
</table>

**Operation: listLoadAverages**

See Operation: getAverageLoad.
Operation: listResourceShares
Scope: Cluster
Deprecated.
List existing resource sharing contexts.
Signature:
listResourceShares()

Table 2-20 listResourceShares

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Operation: listServiceGroups
Scope: Cluster
Displays the names of all Network Gatekeeper service groups. See Administrative Users.
Signature:
listServiceGroups()

Table 2-21 listServiceGroups

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Operation: listServicesInGroup
Scope: Cluster
Displays the names of all Network Gatekeeper services that belong to a specific Network Gatekeeper administrative group. See Administrative Users.

Signature:

\[ \text{listServicesInGroup(group: String)} \]

Table 2-22 listServicesInGroup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>ID of administrative group.</td>
</tr>
</tbody>
</table>

**Operation: listUserInGroup**

Scope: Cluster

Displays the names of all Network Gatekeeper users that belong to a specific Network Gatekeeper administrative group. See Administrative Users.

Signature:

\[ \text{listUserInGroup(group: String)} \]

Table 2-23 listServicesInGroup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>ID of administrative group.</td>
</tr>
</tbody>
</table>

**Operation: listUsers**

Scope: Cluster

Displays a list of all registered administrative users and their corresponding access privileges. See Administrative Users.

Signature:
listUserInGroup()

Table 2-24 listUsers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: login**
Scope: Cluster
Deprecated method. Do not use.

**Operation: removeServiceFromGroup**
Scope: Cluster
Removes a specific Network Gatekeeper service from a given Network Gatekeeper administrative group. See Administrative Users.
Signature:

removeServiceFromGroup(service: String, group: String)

Table 2-25 removeServiceFromGroup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>Service name.</td>
</tr>
<tr>
<td>group</td>
<td>ID of administrative group.</td>
</tr>
</tbody>
</table>
**Operation: removeUserFromGroup**

Scope: Cluster

Removes a specific Network Gatekeeper user from a given Network Gatekeeper administrative group. See Administrative Users.

Signature:

```java
removeUserFromGroup(group: String, user: String)
```

### Table 2-26 removeUserFromGroup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>ID of administrative group.</td>
</tr>
<tr>
<td>user</td>
<td>ID of administrative user.</td>
</tr>
</tbody>
</table>

**Operation: resumeAccess**

Scope: Server

Makes the state of the Network Gatekeeper core transition from SUSPENDED state to RUNNING state.

If invoked when in the RUNNING state nothing will happen.

If invoked in any other state a SLEEException will be thrown.

See Network Gatekeeper Core.

Signature:

```java
resumeAccess()
```
Operation: suspendAccess

Scope: Server

Makes the state of the Network Gatekeeper core transition from RUNNING state to SUSPENDED state.

If invoked when in the SUSPENDED state nothing will happen.

If invoked in any other state a SLEEException will be thrown.

All requests to Network Gatekeeper Core will be rejected with a TRANSIENT exception when in SUSPENDED state.

See Network Gatekeeper Core.

Signature:

suspendAccess()
Managing the States of Services

The following section describes how to control the state of a subset of Network Gatekeeper services, including backwards compatible traffic paths.

- Introduction
- Configuration and Management
- Reference: attributes and operations for SLEE_deployment

Introduction

Backwards compatible software modules installed and run in the Network Gatekeeper are regarded as Network Gatekeeper services. Such services can have one of five states:

- INSTALLED, the service is installed in the SLEE.
- STARTED, the service is started and available for management but cannot send and receive traffic requests.
- ACTIVATED, the service is running and can send and receive traffic requests.
- SUSPENDED, the service is activated but cannot receive new traffic requests. Used for graceful service shutdown.
- ERROR, the service has raised too many critical alarms and has been taken out of service. The allowed number of critical alarms is configurable.
Managing the States of Services

In case of a restart of a Network Gatekeeper server, the services’ restart order and previous operating states are retrieved from the database.

Configuration and Management

To transition a service from state:

- INSTALLED to STARTED, use **Operation: start**
- STARTED to INSTALLED, use **Operation: stop**
- STARTED to ACTIVATED, use **Operation: activate**
- ACTIVATED to STARTED, use **Operation: deactivate**
- ACTIVATED to SUSPENDED, use **Operation: suspend** or **Operation: suspendAllServices**
- SUSPENDED to STARTED, use **Operation: deactivate**
- SUSPENDED to ACTIVATED, use **Operation: resume** or **Operation: resumeAllServices**

Reference: attributes and operations for SLEE_deployment

Below is a list of attributes and operations for configuration and maintenance.

- **Operation: activate**
- **Operation: deactivate**
- **Operation: getServiceActivity**
- **Operation: listServiceActivities**
- **Operation: resume**
- **Operation: resumeAllServices**
- **Operation: start**
- **Operation: stop**
- **Operation: suspend**
- **Operation: suspendAllServices**
**Operation: activate**

Scope: Server

Activates a service.

Signature:

```
activate(service:String)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activate</td>
<td>The name of the service.</td>
</tr>
</tbody>
</table>

**Operation: deactivate**

Scope: Server

Deactivates a service.

Signature:

```
dactivate(service:String)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>The name of the service.</td>
</tr>
</tbody>
</table>

**Operation: getServiceActivity**

Scope: Server

Displays the number of active sessions for a service.

Signature:
Managing the States of Services

```java
getServiceActivity(serviceName:String)
```

### Table 3-3  getServiceActivity

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>The name of the service.</td>
</tr>
</tbody>
</table>

**Operation: listServiceActivities**

Scope: Server
Displays the number of active sessions per service.

Signature:
```java
listServiceActivities()
```

### Table 3-4  listServiceActivities

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: resume**

Scope: Server
Transitions a service from state SUSPENDED to state ACTIVATED.

Signature:
```java
resume(serviceName:String)
```
Operation: resumeAllServices
Scope: Server
Transitions all services from state SUSPENDED to state ACTIVATED.
Signature:
resumeAllServices()

Table 3-5  resumeAllServices

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>The name of the service.</td>
</tr>
</tbody>
</table>

Operation: start
Scope: Server
Transitions a service from state INSTALLED to state STARTED.
Signature:
start(serviceName:String)
Managing the States of Services

Table 3-7  start

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>The name of the service.</td>
</tr>
</tbody>
</table>

**Operation: stop**

Scope: Server

Transitions a service from state STARTED to state INSTALLED.

Signature:

\[ \text{stop}(\text{serviceName}: \text{String}) \]

Table 3-8  stop

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>The name of the service.</td>
</tr>
</tbody>
</table>

**Operation: suspend**

Scope: Server

Transitions a service from state ACTIVATED to state SUSPENDED.

Signature:

\[ \text{stop}(\text{serviceName}: \text{String}) \]
Reference: attributes and operations for SLE deployment

**Table 3-9 stop**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>The name of the service.</td>
</tr>
</tbody>
</table>

**Operation: suspendAllServices**

Scope: Server

Transitions all services from state ACTIVATED to state SUSPENDED.

Signature:

```java
suspendAllServices()
```

**Table 3-10 suspendAllServices**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Managing the States of Services
Managing and Configuring Budgets

The following section explains configuring budgets and describes their relationship to SLA settings.

- Introduction
  - Synchronization of budgets
- Configuration and Management
- Reference: attributes and operations for BudgetService
- Adding a Datasource for the Budget Service

Introduction

In Network Gatekeeper, SLA enforcement is based on budgets maintained by the Budget service. The budget reflects the current traffic request rate based on traffic history. Each Network Gatekeeper server updates both its own local traffic count and the cluster-wide count maintained in one Network Gatekeeper server, the cluster master, based on load and time intervals. The cluster master is, from a cluster-perspective, a singleton service that is highly available and is guaranteed to be available by the WebLogic Server infrastructure. The cluster master is also guaranteed to be active on only one server in the cluster. This ensures accurate SLA enforcement with regards to request counters.

By default, budget quotas are enforced within the cluster. The Budget service is also capable of maintaining budget quotas across domains spread across geographic locations.
Budget values for SLAs that span longer period of time are persisted in the persistent store to minimize the state loss if a cluster master fails.

There are two types of budget caches:

- In-memory only
- In-memory cache backed by persistent storage

When a cluster master is restarted it revives the state from the persistent store. If a cluster master fails, each Network Gatekeeper server continues to independently enforce the SLA accurately to the extent possible, until the role of cluster master has been transferred to an operational server. In such a situation, a subset of the budget cache is lost: the in-memory only budget cache and the parts of the in-memory cache backed by persistent storage that have not been flushed to persistent storage. The flush intervals are configurable, see `Attribute: PersistentBudgetFlushInterval` and `Attribute: PersistentBudgetTimeThreshold`.

A desired accuracy factor for synchronizations can be configured, see `Attribute: AccuracyFactor`.

The configuration settings for these affect accuracy and performance:

- The higher the `Attribute: AccuracyFactor`, the more granularity you have in enforcing the budgets over the time span. This requires more processing power to synchronize the budgets over the cluster.
- The higher the `Attribute: PersistentBudgetFlushInterval` is, the less impact it has on the database performance, and more budget data may be lost in case of server failure.
- The higher the `Attribute: PersistentBudgetTimeThreshold` is, the fewer budgets are likely to be persisted since this value is related to the time intervals for which time limits are defined in the SLAs. A high threshold causes less impact on database performance, but more data may be lost in case of server failure.

### Synchronization of budgets

Budgets are synchronized between all servers in a cluster according to the following algorithm:

\[ r_L = \frac{r}{a \times n} \]

\[ T_L = \frac{T}{a \times n} \]

where:

- \( r_L \) is the slave request count synchronization threshold value.
- \( r \) is a request limit specified in an SLA.
- \( a \) is the accuracy factor, see `Attribute: AccuracyFactor`.
- \( n \) is the number of running WebLogic Network Servers in a cluster.
**Slave intervals**

The request count is the amount of the budget that has been allocated since the last synchronization with the master. The following scenarios are possible:

1. When the request count reaches $r_t$ on a particular node it synchronizes with the master.
2. If the request count does not reach the $r_t$ value and if the count is greater than zero, the slave synchronizes with the master if the time since last synchronization reaches $T_t$.

Synchronization happens as a result of (1) or (2), whichever comes first.

If the request count reaches the threshold value, there will be no explicit synchronization when the timer reaches $T_t$.

Example:

If $r = 10000$, $n = 10$ and $T = 10000$ milliseconds and $a = 1$

$r_t = \frac{10000}{1 \times 10} = 1000$ requests
$T_t = \frac{10000}{1 \times 10} = 1000$ milliseconds

The slave synchronizes with the master if the request count reaches 1000 or if the time since the last synchronization is 1000 ms, whichever comes first, at which point the timer is reset.

**Master Internal**

The master is responsible for enforcing the budget limits across the cluster by keeping track of the request count across all the servers in the cluster.

If there is budget available, the master updates the slaves with the remaining budget whenever the slaves synchronize with the master.

**Failure Conditions**

In the absence of the master, each slave individually enforces the budget limit but caps the requests at $r/n$, thereby guaranteeing that the budget count never reaches the limit.

If the slave fails before it can update the master, the master will not be able to account for that server to enforce SLA. The budget enforcement will be non-operational by at most by $r_t$.

Under certain circumstances, if the master allocates more than the configured budget limit, the budget will be adjusted over time.
Managing and Configuring Budgets

For budgets that span longer period of time, the budget count is persisted in the database to avoid losing all state during master failures. See Attribute: PersistentBudgetTimeThreshold.

**Budget Overrides**

Budgets can have overrides defined in the SLAs, and when a budget is configured with an override, the budget master determines if a given budget is active. If active, it enforces budget limits based on the active override configuration. If overrides are overlapping, no guarantees are provided on which override will be enforced.

**Note:** For an override to be active all of the following must be true:

- Today’s date must be the same as or later than $startDate$
- Today’s date must be earlier than $endDate$ (must not be the same date)
- Current time must be between $startTime$ and $endTime$. If $endTime$ is earlier than $startTime$ the limit spans midnight.
- Current day of week must be between $startDow$ and $endDow$ or equal to $startDow$ or $endDow$. If $endDow$ is less than $startDow$ the limit spans the week end.

### Configuration and Management

Configure the following:

- **Attribute:** PersistentBudgetFlushInterval
- **Attribute:** PersistentBudgetTimeThreshold
- **Attribute:** AccuracyFactor
- **Attribute:** ConfigUpdateInterval

No management operations are available.

**Reference: attributes and operations for BudgetService**

Below is a list of attributes and operations for configuration and maintenance.

- **Attribute:** PersistentBudgetFlushInterval
- **Attribute:** PersistentBudgetTimeThreshold
Adding a Datasource for the Budget Service

System Administrator’s Guide

• **Attribute: AccuracyFactor**

**Attribute: PersistentBudgetFlushInterval**
Scope: Cluster
Format: int
Units: milliseconds
Specifies the time interval between flushes of budgets to persistent storage. See Introduction.

**Attribute: PersistentBudgetTimeThreshold**
Scope: Cluster
Format: int
Units: milliseconds
Specifies threshold value for budgets. Budgets for all time intervals defined in the SLA larger than this value are persisted. See Introduction.

**Attribute: AccuracyFactor**
Scope: Cluster
Format: int
Specifies the accuracy factor. See Introduction.

**Attribute: ConfigUpdateInterval**
Scope: Cluster
Format: int
Unit: milliseconds
Configuration synchronization interval between the slave nodes and the master node.

Adding a Datasource for the Budget Service

Under normal operating conditions, the Budget service makes use of the services of the common transactional (XA) datasource (wlng.datasource) that has been set up for the Network Gatekeeper at large. A datasource is a WLS abstraction that handles connections with the persistent store.
Under very heavy traffic, however, it is possible for the Budget singleton service to be deactivated on all servers. This can happen if the WLS automatic migration mechanism that supports the service becomes “starved” for connections. In this case, a major severity alarm is thrown: Alarm ID 111002, “Budget master unreachable”.

**Note:** Datasource issues are not the only reason this alarm might be thrown.

If you encounter this problem, you can set up a separate singleton datasource for the migration mechanism that will assure that the budget service always has access to the persistent store. This datasource should be configured to use the same database as the common transactional (XA) datasource (wlng.datasource). For more information on singleton service migration in WLS, see the “Migration” chapter in *Using WebLogic Server Clusters*, the Automatic Singleton Service Migration section. Also see the High-availability Database Leasing section for information on the mechanism underlying migration. For information on setting up a separate datasource to support migration of singleton services, like the Budget service, in WLS, see Configuring JDBC Data Sources in *Configuring and Managing WebLogic JDBC*. 
Managing and Configuring EDRs, CDRs and Alarms

The following section describes how to manage and configure EDRs, CDRs, and alarms in Network Gatekeeper.

- About EDRs, CDRs, and Alarms
  - EDR categories
  - EDR format
  - EDRs
  - Alarms
  - CDRs
  - External EDR listeners
- EDRs
  - Configuration of the EDRService
  - Management of the EDRService
  - Reference: attributes and operations for EDRService
- EDR Data
- CDR Data
- Alarm Data
- EDRs and Management of EDRs: Backwards Compatible Traffic Paths
About EDRs, CDRs, and Alarms

Event Data Records (EDRs), are generated in the following ways:

- Automatically using aspects at various location in a network protocol plug-in
- Manually anywhere in the code using the EDRService directly

EDRs are also generated by backwards compatible traffic paths

EDR categories

A new EDR service has been introduced in Network Gatekeeper 3.0. In Network Gatekeeper versions prior to version 3.0, EDRs, alarms and Charging Data Records (CDRs) were three different and unrelated objects. In Network Gatekeeper 3.0, these have been consolidated into one category, EDRs, with alarms and CDRs as subsets of these EDRs.

In order to categorize the objects in the EDR flow as either pure EDRs, alarms or CDRs, the new EDR service uses the following descriptor configuration files:

- edr.xml: this file contains descriptors that describe pure EDRs.
- alarm.xml: this file contains descriptors that describe EDRs that should be considered alarms.
cdr.xml: this file contains descriptors that describe EDRs that should be considered CDRs.

The XML configuration files, together with the XSD edr-config.xsd, are located in APP-INF/classes/edr in the Network Gatekeeper Network tier ear file, which, in the default installation, can be found at $BEA_home/user_projects/domains/<domain name>/wlng_nt.ear.

The descriptor is composed of two parts:

- The <filter> element: this part is the filter
- The <data> element: this part is used to attach additional data to the EDR if it matches the <filter> part

Out-of-the-box, Network Gatekeeper comes with a set of pre-defined descriptor files. Changing and adapting the descriptor files is done as a part of an integration project.

**EDR format**

The following values are always available in the EDR when they are generated from an aspect:

- Class name
- Method name
- Direction (south, north), if the request is travelling from Network Gatekeeper to the network (south) or from the network to Network Gatekeeper (north)
- Position (before, after), if the EDR was emitted before or after the method was invoked or exception was thrown.
- Interface (north, south), if the EDR was emitted from the north interface or from the south interface of the plug-in
- Source (method, exception), if the EDR is related to a method invocation or to an exception.

In addition to these values, the EDR may also contain values relevant to the context of the request.
Below is a description of the content of an EDR. Individual value fields in an EDR are retrieved by name using a key in a name/value pair.

Table 5-1  Content of an EDR

<table>
<thead>
<tr>
<th>String value of name (key) in name/value pair</th>
<th>Valid for Network Gatekeeper version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EdrId</td>
<td>3.0</td>
<td>For enhanced modules this value is found in edr.xml</td>
</tr>
<tr>
<td></td>
<td>2.2</td>
<td>For backwards compatible modules this value is generated.</td>
</tr>
<tr>
<td>ServiceName</td>
<td>3.0</td>
<td>The name, or type, of the service</td>
</tr>
<tr>
<td></td>
<td>2.2</td>
<td></td>
</tr>
</tbody>
</table>
### Table 5-1  Content of an EDR

<table>
<thead>
<tr>
<th>String value of name (key) in name/value pair</th>
<th>Valid for Network Gatekeeper version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerName</td>
<td>3.0</td>
<td>Name of server where the EDR was generated. In Network Gatekeeper 2.2, this was the SleeName.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>3.0</td>
<td>The time at which the EDR was triggered.</td>
</tr>
<tr>
<td></td>
<td>2.2</td>
<td>Milliseconds since midnight, January 1, 1970 UTC.</td>
</tr>
<tr>
<td>ContainerTransactionId</td>
<td>3.0</td>
<td>WebLogic Server transaction ID (if available)</td>
</tr>
<tr>
<td>Class</td>
<td>3.0</td>
<td>Name of the class that logged the EDR</td>
</tr>
<tr>
<td>Method</td>
<td>3.0</td>
<td>Name of the method that logged the EDR</td>
</tr>
<tr>
<td>Direction</td>
<td>3.0</td>
<td>Direction of the request</td>
</tr>
<tr>
<td>Source</td>
<td>3.0</td>
<td>The type of source that logged the EDR</td>
</tr>
<tr>
<td>Position</td>
<td>3.0</td>
<td>Position of the EDR relative to the method that logged the EDR</td>
</tr>
<tr>
<td>Interface</td>
<td>3.0</td>
<td>Interface where the EDR is logged</td>
</tr>
<tr>
<td>Exception</td>
<td>3.0</td>
<td>Name of the exception that triggered the EDR</td>
</tr>
<tr>
<td>SessionId</td>
<td>3.0</td>
<td>Session ID.</td>
</tr>
<tr>
<td>ServiceProviderId</td>
<td>3.0</td>
<td>Service provider account ID.</td>
</tr>
<tr>
<td></td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td>ApplicationId</td>
<td>3.0</td>
<td>Application account ID.</td>
</tr>
<tr>
<td></td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td>AppInstanceGroupId</td>
<td>3.0</td>
<td>Application instance group ID. The username for login purposes.</td>
</tr>
<tr>
<td></td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td>OrigAddress</td>
<td>3.0</td>
<td>The originating address with scheme included. For example tel:1212771234</td>
</tr>
</tbody>
</table>
EDRs fired both by enhanced and backwards compatible traffic paths are passed through the EDRService. All EDRs are dispatched to a JMS distributed topic so external clients can receive them over JMS. All EDRs are also distributed as CORBA requests in order to be backwards compatible with EDR listeners created for Network Gatekeeper 2.2.

This means that EDRs originating from backwards compatible traffic paths and EDRs from enhanced traffic paths are distributed to new listeners, over JMS, and to old listeners, over CORBA. The additional parameters that are available in the new EDR are available in the old EDR, in the additional attributes field.

**Notes:** The CORBA-based EDR listener interface is deprecated. EDRs are not persisted in the database.

### Alarms

Enhanced traffic paths fire EDRs that are mapped to alarms using the alarm xml configuration file: see EDR categories.

Alarms fired by backwards-compatible traffic paths using the SLEE_alarm service are converted to EDRs that follow the same path as an ordinary EDR: see EDRs. As in the case for EDRs, all alarms are also distributed as CORBA requests in order to be backwards compatible with alarm listeners created for Network Gatekeeper 2.2.

<table>
<thead>
<tr>
<th>String value of name (key) in name/ value pair</th>
<th>Valid for Network Gatekeeper version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DestAddress</td>
<td>3.0</td>
<td>The destination address, or addresses, with scheme included. May contain multiple addresses.</td>
</tr>
<tr>
<td>&lt;custom&gt;</td>
<td>3.0</td>
<td>Any additional context-specific information</td>
</tr>
</tbody>
</table>
Notes: The CORBA-based alarm listener interface is deprecated.
Alarms are persisted to the database.

**CDRs**

Enhanced traffic paths fire EDRs that are mapped to CDRs using the cdr.xml configuration file, see [EDR categories](#).

CDRs fired by backwards-compatible traffic paths using the SLEE_charging service are converted to EDRs that follow the same path as an ordinary EDR, see [EDRs](#). As in the case for EDRs, all CDRs are also distributed as CORBA requests in order to be backwards compatible with CDR listeners created for Network Gatekeeper 2.2.

Notes: The CORBA-based CDR listener interface is deprecated.
CDRs are persisted to the database.

**External EDR listeners**

External EDR listeners are simple JMS topic subscribers, see “Creating an EDR listener” on page 31-1 for information on how to create an EDR listener.

**EDRs**

**Configuration of the EDRService**

To configure the behavior of the EDRService, in the managed object EdrService:

1. Specify Attribute: PublishToJMS.
2. Specify Attribute: StoreAlarms.
3. Specify Attribute: StoreCDRs.

**Management of the EDRService**

**Defining batch attributes**

To configure the maximum number of EDRs sent in a batch to a JMS EDR listener and the maximum time to wait before the EDRs in the buffer are sent to listeners:
1. Specify Attribute: `BatchTimeout`.

Reference: attributes and operations for EDRService

**Attribute: BatchTimeout**
Scope: Cluster
Format: int
Unit: milliseconds
Specifies the time out value for a JMS batch.

**Attribute: StatisticsEnabled**
Scope: Cluster
Format: boolean
Specifies if statistics is enabled for EDRService.

**Attribute: BatchSize**
Scope: Cluster
Format: int
Unit: number of EDRs
Specifies the size of the JMS batch.

**Attribute: PublishToJMS**
Scope: Cluster
Format: boolean
Specifies if EDRs shall be published in the JMS topic or not. Needs to be true if external EDR listeners are used.

**Attribute: StoreAlarms**
Scope: Cluster
Format: boolean
Specifies if alarms shall be stored in the database or not.

**Attribute: StoreCDRs**
Scope: Cluster
Format: boolean
Specifies if CDRs shall be stored in the database or not.

**Operation: displayStatistics**
Scope: Cluster
Displays a snapshot of the current statistics for EDRService. The following information is displayed:
- Number of EDRs
- Smallest EDRs message size in bytes
- Biggest EDRs message size in bytes
- Average EDRs message size in bytes
Signature:
```
displayStatistics()
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>displayStatistics</td>
<td></td>
</tr>
</tbody>
</table>

**Operation: resetStatistics**
Scope: Cluster
Resets the statistics for the EDRService.
Signature:
```
resetStatistics()
```
Managing and Configuring EDRs, CDRs and Alarms

Table 5-3  resetStatistics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

EDR Data

See APP-INF/classes/edr/edr.xml in the Network Gatekeeper Network tier ear file, 
$BEA_home/user_projects/domains/<domain name>/wlng_nt.ear.

CDR Data

See APP-INF/classes/edr/cdr.xml in the Network Gatekeeper Network tier ear file, 
$BEA_home/user_projects/domains/<domain name>/wlng_nt.ear.

Alarm Data

See APP-INF/classes/edr/alarm.xml in the Network Gatekeeper Network tier ear file, 
$BEA_home/user_projects/domains/<domain name>/wlng_nt.ear.

EDRs and Management of EDRs: Backwards Compatible Traffic Paths

This section covers management of EDRs fired from backwards compatible traffic paths. These EDRs are also distributed as Network Gatekeeper 3.0 EDRs and any newly created listeners should use the new format for receiving them.

About EDRs for backwards compatible traffic paths

In backwards compatible traffic paths, Event Data Records (EDRs) are forwarded to the SLEE_edr service when certain events occur in Network Gatekeeper. The EDRs originate from
the individual services executing within the Network Gatekeeper. There are a set of predefined
events in each service that can be forwarded to the EDR service.

The EDRs are collected by SLEE_edr, and then distributed to EDR listeners.

The EDR listeners are processes or applications that use CORBA to implement the EdrListener
interface defined in the IDL file SLEEedr.idl. This was a part of the Network Gatekeeper 2.2
integration package.

The CORBA IOR for an EDR listener process is registered in Network Gatekeeper using OAM,
so the EDR listener must be able to print the IOR to file or console.

SLEE_edr allows for filtering on the following parameters:

- EDR ID
- SLEE service
- Service Provider ID
- Application ID
- Address

A filter is defined using the following procedure:

1. An EDR filter is defined using OAM
2. The events to be forwarded to the EDR listeners are defined by adding filter data in the form
   of properties to the filter. The filter data are defined by properties defined as name/value pairs,
   where the following property names are valid:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EdrId</td>
<td>ID of the event</td>
</tr>
<tr>
<td>ServiceName</td>
<td>Service the EDR was fired from.</td>
</tr>
<tr>
<td>ServiceProviderId</td>
<td>Service Provider account ID</td>
</tr>
<tr>
<td>ApplicationId</td>
<td>Application account ID</td>
</tr>
<tr>
<td>Address</td>
<td>Subscriber or terminal address</td>
</tr>
</tbody>
</table>
Several values can be defined for any given name. The names are case-sensitive.

The filter is associated with, or added to, one or more EDR listeners.

When the SLEE_edr receives an incoming event it checks to see if the request matches the filter. In the case of a match, the event is forwarded to the listeners that are associated with the filter.

All filter properties must match, so the filter acts as a white list. When the filter is defined, filter properties in the form of name/value pairs are added. If a certain name property has not been added to the filter, all values for that name are allowed.

In order to make a match, the filter enforcement performs:

- a logical OR between property values defined for a given property name, which means that there must be a matching property value for a given property name.
- a logical AND between property names, which means that there must be matches for all property names defined for a filter.

Since several filters can be added to an EDR listener:

- a logical OR is performed between filters.

Wildcards or regular expressions are not supported for the property values.

For example, if the following properties have been added to a filter:

- Property.name=EdrId, Property.value=1000
- Property.name=EdrId, Property.value=1001
- Property.name=ServiceProviderId, Property.value=SP1

Only EDRs with edrId 1000 or 1001 which originate from Service Provider SP1 are a match.

Since no properties with property names ServiceName, ApplicationId, or Address have been added to the filter, all values for these properties are considered a match for the filter.

**Managing EDRs for backwards compatible traffic paths**

To register an EDR listener, use Operation: registerEdrListener.

To unregister an EDR listener, use Operation: unregisterEdrListener.

To list EDR listeners, use Attribute: EdrListeners (r).

To create an EDR filter, use Operation: createFilter.

To add a filter to an EDR listener, use Operation: addFilterToListener.
To add a property to an EDR filter, use Operation: addFilterProperty.
To remove a property from an EDR filter, use Operation: removeFilterProperty.
To list all EDR filters, use Operation: listFilters.
To list EDR filters associated with a specific EDR listener, use Operation: listFiltersForListener.
To remove an EDR filter from an EDR listener, use Operation: removeFilterFromListener.
To delete an EDR filter, use Operation: deleteFilter.

Configuring the EDR Service

Defining batch attributes
To configure the maximum number of EDRs sent in a batch to an EDR listener and the maximum
time to wait before the EDRs in the buffer are sent to listeners:

1. Specify Attribute: MaxBatchSize.
2. Specify Attribute: MaxNotifyDelay.

Defining EDR listener protection
Listeners that take a long time to process a batch of EDRs before returning can have a significant
negative impact on performance. To protect against this, you can define the maximum time an
EDR listener may execute when receiving a batch of EDRs before returning from the call.
Because a listener that repeatedly exceeds the maximum execution time can be particularly
problematic, you can also define the maximum number of times within a configurable time
interval that the EDR listener may exceed the maximum execution time before that listener is
considered out of service and is removed from the EDR listener distribution list.

1. Specify Attribute: MaxNotifyExecTime
2. Specify Attribute: MaxNoExceedsTimePeriod
3. Specify Attribute: MaxNoExceedsPerTimePeriod

Listing non-responsive EDR listeners
Listeners that are not responding are considered zombies. To see a list of non-responsive listeners
use Operation: listZombieEdrListeners.
Reference: attributes and operations for SLEE_edr

Below is a list of attributes and operations for configuration and maintenance.

- Attribute: MaxNoExceedsPerTimePeriod
- Attribute: MaxNoExceedsTimePeriod
- Attribute: MaxBatchSize
- Attribute: MaxNotifyDelay
- Attribute: ZombieSupervisorAssumeDeadTime
- Attribute: MaxNotifyExecTime
- Operation: addFilterProperty
- Operation: addFilterToListener
- Operation: createFilter
- Operation: deleteFilter
- Operation: listEdrListeners
- Operation: listFilters
- Operation: listFiltersForListener
- Operation: listZombieEdrListeners
- Operation: registerEdrListener
- Operation: unregisterEdrListener
- Operation: removeFilterProperty
- Operation: unregisterEdrListener

**Attribute: MaxNoExceedsPerTimePeriod**
Scope: Cluster
Format: int
Specifies the number of the maximum number of times in during the time period specified in Attribute: MaxNoExceedsTimePeriod an EDR listener may exceed the maximum notification execution time without being removed.

**Attribute: MaxNoExceedsTimePeriod**
Scope: Cluster
Format: int
Unit: seconds
Specifies the time period used in Attribute: MaxNoExceedsPerTimePeriod.

**Attribute: MaxNotifyDelay**
Scope: Cluster
Format: int
Unit: milliseconds.
Specifies the maximum amount of time an EDR is delayed before it is sent to a listener. If the value specified in Attribute: MaxBatchSize is not reached within this time, the EDRs are sent anyway.

**Attribute: EdrListeners (r)**
Scope: Cluster
Displays a list of all registered EDR listeners.

**Attribute: ZombieSupervisorAssumeDeadTime**
Scope: Cluster
Format: long
Unit: milliseconds.
Specifies the maximum time an EDR listener is unreachable before it is assumed dead and is removed.

**Attribute: MaxNotifyExecTime**

Scope: Cluster  
Format: int  
Unit: milliseconds.
Specifies the maximum amount of time an EDR listener is allowed to execute when receiving a batch of EDRs.

**Operation: addFilterProperty**

Scope: Cluster  
Adds an EDR filter property.

Signature:

```java
addFilterProperty(filterId: String, property.name: String, property.value: String)
```

<table>
<thead>
<tr>
<th>Table 5-4 addFilterProperty</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter</strong></td>
</tr>
<tr>
<td>filterId</td>
</tr>
</tbody>
</table>
### Table 5-4 addFilterProperty

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>property.name</code></td>
<td>Name of property to filter on. Applicable properties:</td>
</tr>
<tr>
<td></td>
<td>• EdrId</td>
</tr>
<tr>
<td></td>
<td>• ServiceName</td>
</tr>
<tr>
<td></td>
<td>• ServiceProviderId</td>
</tr>
<tr>
<td></td>
<td>• ApplicationId</td>
</tr>
<tr>
<td></td>
<td>• Address</td>
</tr>
<tr>
<td></td>
<td>If the name is not included as a property, all values for that name are</td>
</tr>
<tr>
<td></td>
<td>considered as matches.</td>
</tr>
<tr>
<td><code>property.value</code></td>
<td>Value of property to filter on.</td>
</tr>
<tr>
<td></td>
<td>For an event to be distributed to the EDR listener, the incoming event</td>
</tr>
<tr>
<td></td>
<td>must match a value given for a name.</td>
</tr>
</tbody>
</table>

### Operation: addFilterToListener

Scope: Cluster

Associates an EDR filter with an EDR listener.

Signature:

```java
addFilterToListener(filterId: String, listenerId: String)
```

### Table 5-5 addFilterToListener

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>filterId</code></td>
<td>ID of the filter.</td>
</tr>
<tr>
<td><code>listenerId</code></td>
<td>ID of the listener.</td>
</tr>
</tbody>
</table>
**Operation: createFilter**

Scope: Cluster

Creates an EDR filter.

Signature:

```java
createFilter(filterId: String)
```

**Table 5-6 createFilter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filterId</td>
<td>ID of the filter.</td>
</tr>
</tbody>
</table>

**Operation: deleteFilter**

Scope: Cluster

Deletes an EDR filter.

Signature:

```java
deleteFilter(filterId: String)
```

**Table 5-7 deleteFilter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filterId</td>
<td>ID of the filter.</td>
</tr>
</tbody>
</table>

**Operation: listEdrListeners**

Scope: Cluster

Displays a list of all registered EDR listeners.

Signature:
listEdrListeners()

Table 5-8 listEdrListeners

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Operation: listFilters

Scope: Cluster

Displays a list of registered EDR filters.

Signature:

listFilters()

Table 5-9 listFilters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Operation: listFiltersForListener

Scope: Cluster

Displays a list of EDR filters associated with a specific EDR listener.

Signature:

listFiltersForListener(filterId: String)
Managing and Configuring EDRs, CDRs and Alarms

Table 5-10  listFiltersForListener

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filterId</td>
<td>ID of the filter.</td>
</tr>
</tbody>
</table>

**Operation: listZombieEdrListeners**

Scope: Cluster

Displays a list of EDR listeners that are not responding and are considered zombies.

Signature:

```
listZombieEdrListeners()
```

Table 5-11  listZombieEdrListeners

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: removeFilterFromListener**

Scope: Cluster

Removes the association between an EDR filter and an EDR listener.

Signature:

```
removeFilterFromListener(filterId: String, listenerId: String)
```
Table 5-12  removeFilterFromListener

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filterId</td>
<td>ID of the filter.</td>
</tr>
<tr>
<td>listenerId</td>
<td>ID of the listener.</td>
</tr>
</tbody>
</table>

**Operation: registerEdrListener**

Scope: Cluster/Server

Registers an EDR Listener.

Signature:

```
registerEdrListener(listenerId: String, listenerIor: String, maxBatchSize: int, registerInAllInstances: byte)
```

Table 5-13  registerEdrListener

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listenerId</td>
<td>ID for the listener.</td>
</tr>
<tr>
<td>listenerIor</td>
<td>CORBA IOR to the EDR listener.</td>
</tr>
<tr>
<td>maxBatchSize</td>
<td>Maximum batch size the listener can handle. There is also a global setting</td>
</tr>
<tr>
<td></td>
<td>for the batch size <strong>Attribute: MaxBatchSize</strong>. The smallest of the values is</td>
</tr>
<tr>
<td></td>
<td>used</td>
</tr>
<tr>
<td>registerInAllInstances</td>
<td>If the listener should be registered in all servers, enter true.</td>
</tr>
<tr>
<td></td>
<td>If the listener should be registered only in the current server, enter false.</td>
</tr>
</tbody>
</table>

**Operation: removeFilterProperty**

Scope: Cluster
Removes an EDR filter property. All criteria must match.

Signature:

```java
removeFilterProperty(filterId: String, property.name: String, property.value: String)
```

### Table 5-14 removeFilterProperty

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filterId</td>
<td>ID of the filter to assign a name/value pair.</td>
</tr>
<tr>
<td>property.name</td>
<td>Name of property to filter on. Applicable properties:</td>
</tr>
</tbody>
</table>
|              | • EdrId
|              | • ServiceName
|              | • ServiceProviderId
|              | • ApplicationId
|              | • Address
|              | If the name is not included as a property, all values for that name are considered as matches. |
| property.value | Value of property to filter on.                                          |
|              | For an event to be distributed to be distributed to the EDR listener, the incoming event must match a value given for a name. |

### Operation: unregisterEdrListener

Scope: Cluster/Server

Unregisters an EDR listener.

Signature:

```java
unregisterEdrListener(listenerId: String)
```
This section covers management of CDRs fired from backwards compatible traffic paths as well as enhanced traffic paths. These CDRs are also distributed as Network Gatekeeper 3.0 CDRs. All newly created listeners should use the new format for receiving CDRs.

**About CDRs**

Backwards compatible Charging Data Records, CDRs, are stored using Network Gatekeeper when certain events occur in Network Gatekeeper. The CDRs originate from the individual services executing within the Network Gatekeeper. There are a set of predefined CDRs in each service that are forwarded to the CDR service.

All management of CDRs is performed using the `SLEE_charging` service in the Network Gatekeeper Management Console. Only a subset of the management attributes and management operations are applicable for Network Gatekeeper 3.0. Deprecated attributes and operations are marked as deprecated.

Backwards compatible CDR listeners are processes or applications that use CORBA to implement the CdrListener interface defined in the IDL file `SLEEcdr.idl`. This file was a part of the Network Gatekeeper 2.2 integration package. The CORBA IOR for a CDR listener process is registered in Network Gatekeeper using OAM, so the CDR listener must be able to print the IOR to file or console.

**Configuring and managing the CDR Service**

**Managing Charging data**

The following operations are related to management of charging data entries:

- Operation: `copyChargingDataToFile`
Operation: countCdrs
Operation: dumpChargingData
Operation: getChargingEntries
Operation: listCdrs

Configuring charging behavior for backwards compatible traffic paths
The following attributes and operations configure the behavior of the backwards compatible charging service itself:

- Attribute: CdrBufferFlushInterval
- Attribute: CdrBufferSizeForAll.
- Operation: setChargingFilter
- Operation: activateCharging
- Operation: deactivateCharging
- Operation: listServiceSettings

Managing backwards compatible external listeners
The following attributes and operations are related to management of external backwards compatible charging listeners:

- Attribute: NrOfChargingEntries (r)
- Operation: addChargingListener
- Operation: isListenerRegistered
- Operation: removeChargingListener

Reference: attributes and operations for SLEE_charging
Below is a list of attributes and operations for configuration and maintenance.

- Attribute: NrOfChargingEntries (r)
- Operation: activateCharging
- Operation: addChargingListener
- Operation: copyChargingDataToFile
- Operation: countCdrs
- Operation: deactivateCharging
- Operation: dumpChargingData
- Operation: getChargingEntries
- Operation: isListenerRegistered
- Operation: listCdrs
- Operation: listServiceSettings
- Operation: removeChargingListener
- Operation: setChargingFilter

**Attribute: NrOfChargingEntries (r)**
Scope: Cluster
Displays the number of CDRs in the database.

**Attribute: CdrBufferFlushInterval**
Scope: Server
Format: int
Unit: seconds
Deprecated. Not applicable for Network Gatekeeper 3.0.
Specifies the maximum time interval between automatic flushes of the CDR buffer into the CDR database.

**Attribute: CdrBufferSizeForAll**
Scope: Server
Format: int
Unit: number of CDR entries.
Deprecated. Not applicable for Network Gatekeeper 3.0.
Specifies the number of CDR entries that are buffered before committing to data base. Increasing this value will improve performance, but the size of this buffer also represents the maximum numbers of CDR entries that might be lost in case of server failure.

The value must be greater than 0.

**Operation: activateCharging**

Scope: Cluster

Deprecated. Only applicable for backwards compatible traffic paths.

Activates `SLEE_charging` for a specific service. Any filter settings for the service are kept.

Signature:

`activateCharging(serviceName: String)`

### Table 5-16 activateCharging

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>serviceName</code></td>
<td>service to activate charging for.</td>
</tr>
</tbody>
</table>

**Operation: addChargingListener**

Scope: Cluster/Server

Registers a charging listener.

Signature:

`addChargingListener(listenerIor: String, registerInAllInstances: boolean)`
Operation: copyChargingDataToFile
Scope: Cluster
Copies the charging data for the specified service provider, application and time period from the charging database table to a file.

Note: Enough disk space must be available to write the data. Each entry in the charging data table may take up to 300 bytes when written to file.

Signature:

```java
copyChargingDataToFile(fileName: String, spId: String, appId: String, fromDate: String, toDate: String)
```
Managing and Configuring EDRs, CDRs and Alarms

Table 5-18  copyChargingDataToFile

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>The name and full path to the file to which the charging data will be exported. The file is automatically created but the directory must exist. The file is saved on a disk connected to a Network Gatekeeper server. If there already is a file with the given name, the process will terminate without exporting the charging data. If the charging data file is very large, a CORBA time-out error may occur. The charging data will be exported to file even if this error occurs.</td>
</tr>
<tr>
<td>Note:</td>
<td>The server must have write privileges in the directory. If MySQL is used, the MySQL server will do the actual writes on one of the database server nodes. In this case the mysql user must have write privileges.</td>
</tr>
<tr>
<td>Note:</td>
<td>If a CORBA time-out error occurs, information on which host the file is stored on will not be displayed. In this case, look for the file on the server on which the database slave is running (MySQL based installations only) and on the server on which the command was performed.</td>
</tr>
<tr>
<td>spId</td>
<td>Service provider account ID.</td>
</tr>
<tr>
<td>appId</td>
<td>Application account ID.</td>
</tr>
<tr>
<td>fromDate</td>
<td>Specifies the start date and time for the interval. Format is YYYY-MM-DD hh:mm:ss</td>
</tr>
<tr>
<td>toDate</td>
<td>Specifies the end date and time for the interval. Format is YYYY-MM-DD hh:mm:ss</td>
</tr>
</tbody>
</table>

**Operation: countCdrs**

Scope: Cluster

Displays the number of CDR entries matching the given criteria.

Signature:
countCdrs(serviceName: String, fromDate: String, toDate: String, completionStatus: int, spAccountId: String, appAccountId: String)

Table 5-19 countCdrs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>Name of source service.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Leave empty to match all.</td>
</tr>
<tr>
<td>fromDate</td>
<td>Specifies the start date and time for the interval.</td>
</tr>
<tr>
<td></td>
<td>Format is YYYY-MM-DD hh:mm:ss</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Leave empty for all up to toDate.</td>
</tr>
<tr>
<td>toDate</td>
<td>Specifies the end date and time for the interval.</td>
</tr>
<tr>
<td></td>
<td>Format is YYYY-MM-DD hh:mm:ss</td>
</tr>
<tr>
<td></td>
<td>Leave empty for all up to current date and time.</td>
</tr>
<tr>
<td>completionStatus</td>
<td>Completion status of the CDR. Use:</td>
</tr>
<tr>
<td></td>
<td>• 0 for Failed</td>
</tr>
<tr>
<td></td>
<td>• 1 for Completed</td>
</tr>
<tr>
<td></td>
<td>• 2 for Partial</td>
</tr>
<tr>
<td></td>
<td>• 3 for Completed but notification failed</td>
</tr>
<tr>
<td></td>
<td>• 4 for Policy rejected</td>
</tr>
<tr>
<td></td>
<td>• -1 for no filtering</td>
</tr>
<tr>
<td>spAccountId</td>
<td>Service provider account ID.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Leave empty to match all.</td>
</tr>
<tr>
<td>appAccountId</td>
<td>Application account ID.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Leave empty to match all.</td>
</tr>
</tbody>
</table>

**Operation: deactivateCharging**

Scope: Cluster
Managing and Configuring EDRs, CDRs and Alarms

Table 5-20 deactivateCharging

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>Name of service.</td>
</tr>
</tbody>
</table>

**Operation: dumpChargingData**

Scope: Cluster

Dumps charging data to file.

**Note:** Dumping the data from the database removes that data from the database. Enough disk space must be available to write the data. Each entry in the charging data table may take up to 300 bytes when written to file.

**Signature:**

dumpChargingData(dumpfileName: String)
### Table 5-21  dumpChargingData

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dumpfileName</td>
<td>The name and full path to the file to which the charging data will be exported. The file is automatically created but the directory must exist. The file is saved on a disk connected to a Network Gatekeeper server. If there already is a file with the given name, the process will terminate without exporting the charging data. If the charging data file is very large, a CORBA time-out error may occur. The charging data will be exported to file even if this error occurs.</td>
</tr>
</tbody>
</table>

**Note:** The server must have write privileges in the directory. If MySQL is used, the MySQL server will do the actual writes on one of the database server nodes. In this case the mysql user must have write privileges.

**Note:** If a CORBA time-out error occurs, information on which host the file is stored on will not be displayed. In this case, look for the file on the server on which the database slave is running (MySQL based installations only) and on the server on which the command was performed.

---

**Operation: getChargingEntries**

**Scope:** Cluster

Displays entries in the charging database.

**Signature:**

```
getChargingEntries(index: int, maxEntries: int)
```
Managing and Configuring EDRs, CDRs and Alarms

Table 5-22  getChargingEntries

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>Start index of list. First index is zero.</td>
</tr>
<tr>
<td>maxEntries</td>
<td>Maximum number of entries to return.</td>
</tr>
</tbody>
</table>

Table 5-23  isListenerRegistered

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listenerIOR</td>
<td>CORBA IOR of charging listener.</td>
</tr>
</tbody>
</table>

Operation: **isListenerRegistered**

Scope: Server

Deprecated. Only applicable for backwards compatible traffic paths and the Network Gatekeeper 2.2 type of charging listeners.

Displays whether a charging listener is registered or not.

Signature:

`isListenerRegistered(listenerIOR: String)`


Operation: **listCdr**

Scope: Server

Displays a list of CDRs filtered on a set of criteria.

Signature:

`listCdr(serviceName: String, fromDate: String, toDate: String, completionStatus: int, spAccountId: String, appAccountId: String, startIndex: long, maxEntries: int)`
## CDRs and Management of CDRs

### System Administrator's Guide 5-33

**Table 5-24 listCdrs**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>Name of source service.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Leave empty to match all.</td>
</tr>
<tr>
<td>fromDate</td>
<td>Specifies the start date and time for the interval. Format is YYYY-MM-DD hh:mm:ss</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Leave empty for all up to toDate.</td>
</tr>
<tr>
<td>toDate</td>
<td>Specifies the end date and time for the interval. Format is YYYY-MM-DD hh:mm:ss</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Leave empty for all up to current date and time.</td>
</tr>
<tr>
<td>completionStatus</td>
<td>Completion status of the CDR. Use:</td>
</tr>
<tr>
<td></td>
<td>- 0 for Failed</td>
</tr>
<tr>
<td></td>
<td>- 1 for Completed</td>
</tr>
<tr>
<td></td>
<td>- 2 for Partial</td>
</tr>
<tr>
<td></td>
<td>- 3 for Completed but notification failed</td>
</tr>
<tr>
<td></td>
<td>- 4 for Policy rejected</td>
</tr>
<tr>
<td></td>
<td>- -1 for no filtering</td>
</tr>
<tr>
<td>spAccountId</td>
<td>Service provider account ID.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Leave empty to match all.</td>
</tr>
<tr>
<td>appAccountId</td>
<td>Application account ID.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Leave empty to match all.</td>
</tr>
<tr>
<td>startIndex</td>
<td>Start index of list. First index is zero.</td>
</tr>
<tr>
<td>maxEntries</td>
<td>Maximum number of entries to return.</td>
</tr>
</tbody>
</table>

**Operation: listServiceSettings**

Scope: Server
Managing and Configuring EDRs, CDRs and Alarms

Deprecated. Not applicable for Network Gatekeeper 3.0.
Displays a list of charging related attributes per service.
Attributes, of type boolean, included per service:

- **Active** if \texttt{SLEE\_charging} is active for the service.
- **DB** if \texttt{SLEE\_charging} persists the CDRs to database.
- **Charge completed** if \texttt{SLEE\_charging} generates CDRs for requests that are considered complete.
- **Charge failed** if \texttt{SLEE\_charging} generates CDRs for requests that are considered failed.
- **Charge partial** if \texttt{SLEE\_charging} generates CDRs for requests that are considered partly completed.

Signature:

\texttt{listServiceSettings()}

\begin{table}[h]
\centering
\begin{tabular}{ll}
\hline
Parameter & Description \\
\hline
- & - \\
\hline
\end{tabular}
\caption{listServiceSettings}
\end{table}

**Operation: removeChargingListener**

Scope: Cluster/Server

Deprecated. Only applicable for backwards compatible traffic paths and the Network Gatekeeper 2.2 type of charging listeners.

Removes an external charging listener.

Signature:

\texttt{removeChargingListener(listenerIOR: String, removeFromAllInstances: boolean)}
Table 5-26  removeChargingListener

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listenerIOR</td>
<td>The IOR to an object implementing the alarm listener interface.</td>
</tr>
<tr>
<td></td>
<td>String format</td>
</tr>
<tr>
<td>removeFromAllInstances</td>
<td>If set to TRUE the listener will be removed from all instances in the domain.</td>
</tr>
<tr>
<td></td>
<td>If set to FALSE the listener will only be removed from the current server.</td>
</tr>
</tbody>
</table>

**Operation: setChargingFilter**

Scope: Cluster

Deprecated. Only applicable for backwards compatible traffic paths.

Specifies, for each service, based on the transaction result, which transactions should be stored in the charging database and distributed to external charging listeners.

Signature:

```
setChargingFilter(serviceName: String, completed: boolean, partial: boolean, failed: boolean)
```

Table 5-27  setChargingFilter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>Service name.</td>
</tr>
<tr>
<td>completed</td>
<td>Specifies if CDRs with completion status COMPLETED should be logged.</td>
</tr>
<tr>
<td>partial</td>
<td>Specifies if CDRs with completion status PARTIAL should be logged.</td>
</tr>
<tr>
<td>failed</td>
<td>Specifies if CDRs with completion status FAILED should be logged.</td>
</tr>
</tbody>
</table>
Alarms and Management of Alarms

This section covers management of alarms emitted from backwards compatible traffic paths and enhanced traffic paths. Alarms emitted from backwards compatible traffic paths are also distributed as Network Gatekeeper 3.0 alarms and any newly created listeners should use the new format for receiving alarms.

About alarms

Managing alarms covers three areas of administration:

- Changing the severity level on individual alarms - see Changing alarm severity levels for backwards compatible traffic paths.

- Viewing the list of raised alarms and deleting them from the database - see Managing raised alarms.

- Adding alarm listeners - see Managing external alarm listeners for backwards compatible traffic paths.

- For information on understanding and responding to individual alarms, see Handling Alarms - BEA WebLogic Network Gatekeeper, a separate document in the WebLogic Network Gatekeeper documentation set.

Raised alarms are stored in the database, emitted to alarm listeners, and distributed to registered SNMP listeners.

Only a subset of the management attributes and management operations are applicable for Network Gatekeeper 3.0. Deprecated attributes and operations are marked as deprecated.

Managing alarms

Managing raised alarms

To view raised alarms, use Operation: listAlarms.

To delete alarm entries from the database, use Operation: deleteAlarms.

Changing alarm severity levels for backwards compatible traffic paths

To change an alarm’s severity level, use Operation: setReconfiguredAlarmSeverity

To list alarms whose severity levels have been changed, use Operation: listReconfiguredAlarmSeverities
To restore the default severity level of an alarm, use Operation: removeReconfiguredAlarmSeverity.

Managing external alarm listeners for backwards compatible traffic paths

A backwards compatible alarm listener must implement the interface defined in AlarmService.idl. This CORBA interface definition file was a part of Network Gatekeeper 2.2 integration module. The CORBA IOR for the listener must be registered in Network Gatekeeper.

To add (register) an alarm listener, use Operation: addAlarmListener.

To remove (unregister) an alarm listener, use Operation: removeAlarmListener.

To check if an alarm listener is registered, use Operation: isListenerRegistered.

For new alarm listeners, see EDRs.

Reference: attributes and operations for SLEE_alarm

Below is a list of attributes and operations for configuration and maintenance.

- Attribute: AlarmBroadcastInterval
- Attribute: AlarmBroadcasterIOR (r)
- Attribute: FilterLevel (r)
- Operation: addAlarmListener
- Operation: countAlarmInfo
- Operation: deleteAlarm
- Operation: deleteAlarms
- Operation: getAlarm
- Operation: getReconfiguredAlarmSeverity
- Operation: isListenerRegistered
- Operation: listAlarmInfo
- Operation: listAlarms
- Operation: listReconfiguredAlarmSeverities
• **Operation: removeAlarmListener**
• **Operation: removeReconfiguredAlarmSeverity**
• **Operation: setFilterLevel**
• **Operation: setReconfiguredAlarmSeverity**

**Attribute: AlarmBroadcastInterval**
Scope: Server
Format: int
Unit: seconds
Deprecated. Only valid for Network Gatekeeper 2.2 type of alarm listeners.
Specifies the time interval between alarm broadcasts.

**Attribute: AlarmBroadcasterIOR (r)**
Scope: Server
Deprecated. Only valid for Network Gatekeeper 2.2 type of alarm listeners.
Displays the CORBA IOR for the alarm broadcaster.

**Attribute: FilterLevel (r)**
Read-only.
Scope: Server
Format: int
Deprecated. Only valid for backwards compatible traffic paths.
Specifies which alarms are logged.
The active filter level is represented with a digit:
• 1 to log all alarms.
• 2 to log minor, major and critical alarms
• 3 to log major and critical alarms
• 4 to log only critical alarms
**Operation: addAlarmListener**

Scope: Cluster/Server

Deprecated. Only valid for Network Gatekeeper 2.2 type of alarm listeners.

Adds an external alarm listener.

Signature:

```java
addAlarmListener(listenerIOR: String, registerInAllInstances: boolean)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listenerIOR</td>
<td>The IOR to an object implementing the alarm listener interface. String format</td>
</tr>
<tr>
<td>registerInAllInstances</td>
<td>If set to TRUE the listener will be registered with all alarm service instances in the domain. If set to FALSE the alarm listener will only be registered with the instance in the current server. That is, only alarms raised by services in the current Network Gatekeeper will be provided</td>
</tr>
</tbody>
</table>

**Operation: countAlarmInfo**

Scope: Cluster

Displays the number of emitted alarms matching the given criteria.

Signature:

```java
countAlarmInfo(alarmId: int, severity: int, fromDate: String, toDate: String)
```
Managing and Configuring EDRs, CDRs and Alarms

### Table 5-29  countAlarmInfo

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarmId</td>
<td>Alarm identifier. Use 0 to retrieve all alarms without filtering on identifier.</td>
</tr>
<tr>
<td>severity</td>
<td>The minimum alarm severity to be included in the listing. Use: • 1 for WARNING • 2 for MINOR • 3 for MAJOR • 4 for CRITICAL</td>
</tr>
<tr>
<td>fromDate</td>
<td>Start date and time. Format YYYY-MMM-DD HH:MM:SS (24-hour notation)</td>
</tr>
<tr>
<td>toDate</td>
<td>End date and time. Format YYYY-MMM-DD HH:MM:SS (24-hour notation)</td>
</tr>
</tbody>
</table>

**Note:** Leave empty for all from fromDate to current date and time.

---

**Operation: deleteAlarm**

**Scope: Cluster**

Deletes an alarm with a given ID from the database.

**Signature:**

`deleteAlarm(alarmId:long)`
### Operation: deleteAlarms

**Scope:** Cluster

Deletes a range of alarms matching the given criteria from the database.

**Signature:**

```java
deleteAlarms(serviceName: String, severity: int, fromDate: String, toDate: String, identifier: int)
```

### Table 5-30 deleteAlarm

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarmId</td>
<td>ID of alarm.</td>
</tr>
</tbody>
</table>

### Table 5-31 deleteAlarms

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>Name of service that emitted the alarm.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Leave empty for alarms from all services.</td>
</tr>
<tr>
<td>severity</td>
<td>Severity level to filter on:</td>
</tr>
<tr>
<td></td>
<td>0 - all levels</td>
</tr>
<tr>
<td></td>
<td>1 - warning</td>
</tr>
<tr>
<td></td>
<td>2 - minor</td>
</tr>
<tr>
<td></td>
<td>3 - major</td>
</tr>
<tr>
<td></td>
<td>4 - critical</td>
</tr>
<tr>
<td>fromDate</td>
<td>Start date and time.</td>
</tr>
<tr>
<td></td>
<td>Format YYYY-MMM-DD HH:MM:SS (24-hour notation)</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Leave empty for all up to toDate.</td>
</tr>
</tbody>
</table>
Managing and Configuring EDRs, CDRs and Alarms

**Table 5-31 deleteAlarms**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| **toDate** | End date and time.  
Format YYYY-MM-DD HH:MM:SS (24-hour notation)  
**Note:** Leave empty for all from fromDate to current date and time. |
| **identifier** | Alarm type number to filter on.  
Use 0 for all alarm types. |

**Operation: getAlarm**

Scope: Cluster

Displays information for an alarm with a certain ID.

Signature:

```
getAlarm(alarmId: long)
```

**Table 5-32 getAlarm**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>alarmId</strong></td>
<td>ID of alarm.</td>
</tr>
</tbody>
</table>

**Operation: getReconfiguredAlarmSeverity**

Scope: Cluster

Deprecated. Only valid for backwards compatible traffic paths.

Displays the reconfigured severity for the specified alarm identifier.

Signature:

```
getReconfiguredAlarmSeverity(identifier: int)
```
Table 5-33  getReconfiguredAlarmSeverity

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>identifier</td>
<td>Alarm type identifier.</td>
</tr>
</tbody>
</table>

**Operation: isListenerRegistered**

Scope: Server

Deprecated. Only valid for Network Gatekeeper 2.2 type of alarm listeners.

Checks if an alarm listener matching the criteria is registered.

Signature:

```java
isListenerRegistered(listenerIOR: String)
```

Table 5-34  isListenerRegistered

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listenerIOR</td>
<td>IOR of listener.</td>
</tr>
<tr>
<td></td>
<td>String format.</td>
</tr>
</tbody>
</table>

**Operation: listAlarmInfo**

Scope: Cluster

Displays a list of raised alarms matching the give criteria.

Signature:

```java
listAlarmInfo(alarmId: int, severity: int, fromDate: String, toDate: String, startIndex: long, maxEntries: int)
```
Table 5-35 listAlarmInfo

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarmId</td>
<td>ID of alarm type.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Leave empty for all alarm types</td>
</tr>
<tr>
<td>severity</td>
<td>Severity level to filter on:</td>
</tr>
<tr>
<td></td>
<td>1 - warning</td>
</tr>
<tr>
<td></td>
<td>2 - minor</td>
</tr>
<tr>
<td></td>
<td>3 - major</td>
</tr>
<tr>
<td></td>
<td>4 - critical</td>
</tr>
<tr>
<td>fromDate</td>
<td>Start date and time.</td>
</tr>
<tr>
<td></td>
<td>Format YYYY-MMM-DD HH:MM:SS (24-hour notation)</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Leave empty for all up to toDate.</td>
</tr>
<tr>
<td>toDate</td>
<td>End date and time.</td>
</tr>
<tr>
<td></td>
<td>Format YYYY-MMM-DD HH:MM:SS (24-hour notation)</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Leave empty for all from fromDate to current date and time.</td>
</tr>
<tr>
<td>startIndex</td>
<td>The first entry within the matching result to return</td>
</tr>
<tr>
<td>maxEntries</td>
<td>The maximum number of entries to return</td>
</tr>
</tbody>
</table>

**Operation: listAlarms**

**Scope:** Cluster

Displays a list of raised alarms matching the give criteria.

**Signature:**

```java
listAlarms(serviceName: String, severity: int, fromDate: String, toDate: String, identifier: int)
```
**Table 5-36  listAlarms**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| serviceName | Name of service that emitted the alarm.  
  **Note:** Leave empty for alarms from all services. |
| severity | Severity level to filter on:  
  1 - warning  
  2 - minor  
  3 - major  
  4 - critical |
| fromDate | Start date and time.  
  Format YYYY-MMM-DD HH:MM:SS (24-hour notation)  
  **Note:** Leave empty for all up to toDate. |
| toDate | End date and time.  
  Format YYYY-MMM-DD HH:MM:SS (24-hour notation)  
  **Note:** Leave empty for all from fromDate to current date and time. |
| identifier | Alarm type number to filter on.  
  Use 0 for all alarm types. |

**Operation: listReconfiguredAlarmSeverities**

**Scope:** Cluster

Deprecated. Only valid for backwards compatible traffic paths.

Displays a list of all alarms that have a reconfigured severity level.

**Signature:**

```plaintext
listReconfiguredAlarmSeverities()
```
Table 5-37  listReconfiguredAlarmSeverities

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Operation: listValidSeverities**

Scope: Cluster
Displays the valid severity values.

Signature:

listValidSeverities()

Table 5-38  listValidSeverities

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Operation: removeAlarmListener**

Scope: Cluster/Server
Deprecated. Only valid for Network Gatekeeper 2.2 type of alarm listeners.
Removes an external alarm listener.

Signature:

removeAlarmListener(listenerIOR:String, removeFromAllInstances: boolean)
Table 5-39 addAlarmListener

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listenerIOR</td>
<td>The IOR to an object implementing the alarm listener interface.</td>
</tr>
<tr>
<td></td>
<td>String format</td>
</tr>
<tr>
<td>removeFromAllInstances</td>
<td>If set to TRUE the listener will be removed from all instances in the domain.</td>
</tr>
<tr>
<td></td>
<td>If set to FALSE the listener will only be removed from the current server.</td>
</tr>
</tbody>
</table>

Table 5-40 removeReconfiguredAlarmSeverity

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>identifier</td>
<td>Alarm type identifier.</td>
</tr>
</tbody>
</table>

Operation: removeReconfiguredAlarmSeverity

Scope: Cluster

Deprecated. Only valid for backwards compatible traffic paths.

Resets the severity level of an alarm type with a previously reconfigured severity level.

Signature:

removeReconfiguredAlarmSeverity(identifier: int)

Table 5-41 removeReconfiguredAlarmSeverity

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>identifier</td>
<td>Alarm type identifier.</td>
</tr>
</tbody>
</table>

Operation: setFilterLevel

Scope: Cluster

Deprecated. Set filter level for alarms.

Signature:

setFilterLevel(level: int)
Managing and Configuring EDRs, CDRs and Alarms

Table 5-42  setFilterLevel

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>level</td>
<td>Set filter level for the alarm service.</td>
</tr>
<tr>
<td>WARNING</td>
<td>1</td>
</tr>
<tr>
<td>MINOR</td>
<td>2</td>
</tr>
<tr>
<td>MAJOR</td>
<td>3</td>
</tr>
<tr>
<td>CRITICAL</td>
<td>4</td>
</tr>
</tbody>
</table>

Alarms with severity lower than current filter setting will be discarded.

Operation: setReconfiguredAlarmSeverity

Scope: Cluster

Deprecated. Only valid for backwards compatible traffic paths.

Resets the severity level of an alarm type with a previously reconfigured severity level.

Signature:

setReconfiguredAlarmSeverity(identifier: int, severity: int)

Table 5-43  setReconfiguredAlarmSeverity

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>identifier</td>
<td>Alarm type identifier.</td>
</tr>
<tr>
<td>severity</td>
<td>New severity level.</td>
</tr>
</tbody>
</table>
Managing and Configuring Statistics and Transaction Licenses

The following sections describe the statistics functionality and the operation and maintenance procedures for statistics:

- About Statistics Generation and Reports
- Overview of Statistics Reports
  - System report to console
  - Report to file
  - Weekly system report
  - License limit log report
- Operation, Maintenance, and Provisioning
  - Reference: attributes and operations for SLEE_statistics
- Transaction Types
- Transaction Types for Enhanced Traffic Paths
- Transaction Types for Backwards Compatible Traffic Paths

About Statistics Generation and Reports

WebLogic Network Gatekeeper keeps usage statistics in terms of the number of transactions handled over time. Transactions are grouped into transaction types. Transaction types are used
for transaction limit enforcement and for grouping reports. Transaction types are in turn categorized into different categories for license enforcement.

Statistics are applied only to traffic paths. Verification mechanisms ensure that all network protocol plug-ins have the statistics aspects applied. This verification takes place when the plug-in is deployed into Network Gatekeeper.

Enhanced traffic paths use aspects to generate statistics. Backwards compatible plug-ins use other mechanisms. However, in both cases, statistics are generated and included in reports.

Statistics are held in an in-memory store and flushed to database at a given time interval. Statistics reports are created based on information in the database.

It is possible to get a snapshot of the current status of the transaction, or request, counters. This information is fetched from an in-memory store.

These report types are available:

- Overview of Statistics Reports
- Weekly system report
- License limit log report

**Overview of Statistics Reports**

Statistics are used to generate reports filtered on a number of parameters:

- Time interval
  - Start time
  - End time
- Individual statistics types or an aggregate of all statistics types.
- Originator of the transaction:
  - Service provider account ID
  - Application account ID
- Per cluster or per server

**Note:** All combinations of the above are not supported.
System report to console
This report is created by Operation: getSystemStatistics. The output is presented in the console.

Report to file
This report is created by Operation: saveStatisticsToFile. The format is adapted for programmatic processing of the file.

Weekly system report
The weekly report is a pre-defined report. It shows the total number of transactions through Network Gatekeeper hour by hour during a specified week. It also shows total usage for each day and the average transaction rate (transactions/second) during the busy hour of each day. The busy hour is defined as the 60 minutes during which the largest number of transactions are handled, and does not depend on clock hours. Any 60 minute period (5 minute intervals are used) can be identified as the busy hour.

A weekly system statistics report shows:
- The total number of transactions during the specified week
- The number of transactions during each hour of the days in the week
- The number of transactions during each day of the week
- The transaction rate (transactions/second) during the busy hour of each day

This report is stored on file.

License limit log report
A license limit log can be extracted from the Network Gatekeeper. The log file contains a set of entries, where each entry represents the average transactions per second during the busy hour of a 24-hour period starting at 12:00 AM and ending 11:59 PM the previous day.

The license limit log report is an XML file with a header and a footer.

Listing 6-1   Structure of license limit log file

```xml
<transaction_limit_log>
  <start> </start>
</transaction_limit_log>
```
All information is contained within the tags `<transaction_limit_log>`.

A header, encapsulated by the tag `<header>`, contains information on the time period over which the log is generated, with a start and end date in the tags `<start>` and `<end>`, respectively. The format is DD-MM-YYYY.

Directly following the tag `<end>`, one or more log entries are found in the tag `<log_entry>`.

There is one `<log_entry>` created for each day and transaction type.

This tag contains a set of attributes:

- `group`: the transaction group for which it is valid. Possible values are Platform or BEA-modules.
- `start`: the start date and time for the busy hour. Format is YYYY-MM-DD HH:MM, where HH is given in 24 hour format.
- `end`: the end date and time for the busy hour. Format is YYYY-MM-DD HH:MM, where HH is given in 24 hour format.
- `tps`: the average transactions per second during the busy hour.
- `limit`: the average transactions per second allowed according to the license file.
- `exceeded`: true if the limit was exceeded. False otherwise.

A checksum is contained in the tag `<checksum>`. This tag contains a checksum created based on the content of the file. The checksum is used for validating that the file has not been changed.
Listing 6-2  License limit log file example.

```xml
<transaction_limit_log>
<start>2006-01-01</start>
<end>2006-01-31</end>
<log_entry group="BEA-modules" start="2007-01-01 13:45" end="2007-01-01 14:45"
 tps="27.35" limit="50" exceeded="false"/>
<checksum>f8b904410896b3f92159524c6c68</checksum>
</transaction_limit_log>
```

Operation, Maintenance, and Provisioning

Below is a summary of operations, by category.

List of Attributes:

- Attribute: CounterSnapshot (r)
- Attribute: CounterSnapshotAll (r)
- Attribute: SiteId
- Attribute: StatisticsStatus (r)
- Attribute: StoreInterval (r)

List of operations:

- Operation: addStatisticType
- Operation: createLicenseLimitLog
- Operation: createWeeklyReport
- Operation: getStatistics
- Operation: getStatisticsAppID
- Operation: getSystemStatistics
- Operation: listStatisticTypeDescriptors
Managing and Configuring Statistics and Transaction Licenses

Operation: listStatistics
Operation: listStatisticTypes
Operation: removeStatisticType
Operation: saveStatisticsToFile

Reference: attributes and operations for SLEE_statistics

Attributes and operations
Below is a list of attributes and operations for configuration and maintenance.

Attribute: SiteId
Scope: Cluster
Specifies an ID that will be included in statistics report generated.

Attribute: CounterSnapshotAll (r)
Read only.
Scope: Cluster
Displays a snapshot with information about all statistics-related counters in all instances of Network Gatekeeper servers.
The information is useful for analyzing runtime traffic and can be used to get an approximate value of the traffic throughput by invoking it periodically.

Note: The counters are reset periodically.

Attribute: CounterSnapshot (r)
Read only.
Scope: Server
Displays a snapshot with information about all statistics-related counters for the chosen Network Gatekeeper server.
The information is useful for analyzing runtime traffic and can be used to get an approximate value of the traffic throughput by invoking it periodically.

Note: The counters are reset periodically.
**Attribute: StoreInterval (r)**
Read only.
Unit: seconds.
Scope: Cluster
Displays information on how often statistics data are written to the database.

**Operation: addStatisticType**
Scope: Cluster
Adds a statistic type. Used in conjunction with Extension Traffic Paths.
Signature:
```
addStatisticType(id: int, name: String, weight: int)
```

**Table 6-1  addStatisticType**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Statistic type ID.</td>
</tr>
<tr>
<td>name</td>
<td>Statistic type name.</td>
</tr>
<tr>
<td>weight</td>
<td>Unused. A value must be provided.</td>
</tr>
</tbody>
</table>

**Operation: createLicenseLimitLog**
Scope: Cluster
Creates start and end date limits for log.
Signature:
```
createLicenseLimitLog(filename: String, startDate: String, endDate: int)
```
**Operation: createWeeklyReport**

**Scope:** Cluster

Creates a weekly report. See Weekly system report for a description of the report.

**Signature:**

```java
createWeeklyReport(entOpID : String, startDate: String, fileName: String, decimals: int)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| filename  | The filename for the report. The file is created on the local file system of the selected server. Must include an absolute path.  
**Note:** The file must not already exist. If it does already exist, the method will fail. |
| startDate | Specifies the start date for the report. Format is YYYY-MM-DD. |
| endDate   | Specifies the end date for the report. Format is YYYY-MM-DD. |
| entOpId   | Service provider ID to filter on.  
**Note:** Leave empty to include all service provider account in the report. |
| startDate | Specifies the start date for the report. Format is YYYY-MM-DD. |
Table 6-3 createWeeklyReport

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| filename  | The filename for the report. Must include an absolute path. The file is created on the local file system of the selected server.  
**Note:** The file must not already exist. If it does already exist, the method will fail. |
| decimals  | Number of decimals in entry for transactions/second in the report. |

**Operation: getStatistics**

Scope: Cluster or Server  
Used internally. Not rendered in console.

Signature:  
get(sleeName: String, statisticsType: String)

Table 6-4 getStatistics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| sleeName    | Specifies the name of the server whose generated statistics you wish to display.  
**Note:** Leave empty to display statistics generated in all servers. |
| statisticsType | Statistics type ID for the statistics type to display.  
**Note:** Use -1 to display all statistics types. |
Managing and Configuring Statistics and Transaction Licenses

Table 6-4 getStatistics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fromDate</td>
<td>Specifies the start date and time for the interval to display. Format is YYYY-MM-DD hh:mm. Note: Leave empty to display all statistics up to the date and time specified in toDate.</td>
</tr>
<tr>
<td>toDate</td>
<td>Specifies the end date and time for the interval to display. Format is YYYY-MM-DD hh:mm. Note: Leave empty to display all statistics generated from the date and time specified in fromDate until current date and time.</td>
</tr>
</tbody>
</table>

Operation: getStatisticsAppID

Scope: Cluster or Server
Not rendered in console. Used internally.

Signature:

getStatisticsAppID(sleeName: String, statisticsType: int, fromDate: String, toDate: String)
**Operation: getSystemStatistics**

Scope: Cluster or Server

Displays a summary of system statistics for the last minute(s). See Overview of Statistics Reports for information on output format.

Signature:

```plaintext
getSystemStatistics(minutes: int)
```

---

**Table 6-5 getStatisticsAppID**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fromDate</td>
<td>Specifies the start date and time for the interval to display. Format is YYYY-MM-DD hh:mm. Note: Leave empty to display all statistics up to the date and time specified in toDate.</td>
</tr>
<tr>
<td>toDate</td>
<td>Specifies the end date and time for the time interval. Format is YYYY-MM-DD hh:mm. Note: Leave empty to generated statistics from the date and time specified in fromDate until current date and time.</td>
</tr>
<tr>
<td>entOpID</td>
<td>Service provider ID to filter on.</td>
</tr>
<tr>
<td>clientAppID</td>
<td>Application ID to filter on.</td>
</tr>
</tbody>
</table>

Note: Leave empty to display all statistics for all applications for the service provider ID specified in entOpID.

---

**Table 6-6 getSystemStatistics**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note: minutes</td>
<td>The number of minutes relative to the current time.</td>
</tr>
</tbody>
</table>
### Operation: `listStatisticTypeDescriptors`

**Scope:** Cluster

Displays a list of all available statistics type descriptors. The descriptor contain information on `transactionTypeName` and `transactionTypeID`.

**Signature:**

```java
listStatisticTypeDescriptors()
```

### Table 6-7 `listStatisticTypeDescriptors`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Operation: `listStatisticTypes`

**Scope:** Cluster

Specifies the statistics types included in the statistics reports.

**Signature:**

```java
listStatisticTypes()
```

### Table 6-8 `listStatisticTypes`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Operation: `listStatistics`

**Scope:** Cluster

Gets statistics data stored in the database. Can be filtered on:

- service provider ID
- Application ID
  Not rendered in console. Used internally.

Signature:

```java
listStatistics(statisticsType: int, fromDate: long, toDate: long, spAccountId: String, appAccountId: String)
```

### Table 6-9 listStatistics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>statisticsType</code></td>
<td>ID for the statistics type. Use -1 to display all statistics types.</td>
</tr>
<tr>
<td><code>fromDate</code></td>
<td>Specifies the start date and time for the time interval. Given in number of milliseconds since the UNIX epoch. Use negative value for all statistics up to the date and time specified in <code>toDate</code>.</td>
</tr>
<tr>
<td><code>toDate</code></td>
<td>Specifies the end date and time for the time interval. Given in number of milliseconds since the UNIX epoch. Use negative value for statistics generated from the date and time specified in <code>fromDate</code> until current date and time.</td>
</tr>
<tr>
<td><code>spAccountId</code></td>
<td>Service provider ID to filter on. Leave empty for all.</td>
</tr>
<tr>
<td><code>appAccountId</code></td>
<td>Application ID to filter on. Leave empty for all.</td>
</tr>
</tbody>
</table>

**Operation: removeStatisticType**

Scope: Cluster

Removes a statistics type.

Must not be used.
Managing and Configuring Statistics and Transaction Licenses

Signature:

removeStatisticType(statisticsType: int)

Table 6-10 listStatistics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>statisticsType</td>
<td>ID for the statistics type.</td>
</tr>
</tbody>
</table>

**Operation: saveStatisticsToFile**

Scope: Cluster/Server

Saves a statistics report to file.

Signature:

saveStatisticsToFile(sleeName: String, filename: String, startDate: String, endDate: int)

Table 6-11 saveStatisticsToFile

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sleeName</td>
<td>Specifies the name of the server.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Leave empty to save statistics generated in all servers.</td>
</tr>
<tr>
<td>statisticsType</td>
<td>ID for the statistics type.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Use -1 to display all statistics types.</td>
</tr>
<tr>
<td>fromDate</td>
<td>Specifies the start date and time for the time interval.</td>
</tr>
<tr>
<td></td>
<td><strong>Format is YYYY-MM-DD hh:mm.</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Leave empty for all statistics up to the date and time specified in toDate.</td>
</tr>
</tbody>
</table>
### Transaction Types

The following transaction types are defined:

- TRANSACTION_TYPE_USER_LOCATION
- TRANSACTION_TYPE_USER_STATUS
- TRANSACTION_TYPE_USER_INTERACTION_RESPONSE
- TRANSACTION_TYPE_USER_INTERACTION_NO_RESPONSE
- TRANSACTION_TYPE_MESSAGING_SEND
- TRANSACTION_TYPE_MESSAGING_RECEIVE
- TRANSACTION_TYPE_CALL_CONTROL_NETWORK_INITIATED.
- TRANSACTION_TYPE_CALL_CONTROL_SERVICE_INITIATED.
- TRANSACTION_TYPE_CHARGING_DIRECT
- TRANSACTION_TYPE_CHARGING_RESERVED
- TRANSACTION_TYPE_OTHER
- TRANSACTION_TYPE_SUBSCRIBER_PROFILE

---

**Table 6-11 saveStatisticsToFile**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>toDate</td>
<td>Specifies the end date and time for the time interval. Format is YYYY-MM-DD hh:mm.</td>
<td>Leave empty for all statistics generated from the date and time specified in fromDate until current date and time.</td>
</tr>
<tr>
<td>filename</td>
<td>Specifies the filename for the report. Must include an absolute path. The file is created on the local file system of the selected server.</td>
<td>The file must not already exist. If it does already exist, the method will fail.</td>
</tr>
</tbody>
</table>
Which events generate which statistics for a certain transaction type are described in Transaction Types for Enhanced Traffic Paths and Transaction Types for Backwards Compatible Traffic Paths.

The transaction type TRANSACTION_TYPE_EXTENSION is used for new traffic paths, developed as extensions to Network Gatekeeper.
Transaction Types for Enhanced Traffic Paths

Table 6-12 outlines the correlation between methods being invoked from either an application or the telecom network and the corresponding transaction type for enhanced traffic paths.

For interfaces where Network Gatekeeper implements the Web Service, the method is defined in the application-facing interface. Statistics counters related to the transaction type are updated regardless of which plug-in the request is passed on to.

For interfaces where Network Gatekeeper is the consumer of the Web Service, the method is the method being invoked as a result of a protocol-dependant request from the network. Statistics counters related to the transaction type are updated depending on which plug-in the request arrives at.

<table>
<thead>
<tr>
<th>Method</th>
<th>Transaction type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parlay X 2.1 Part 2: Third Party Call, interface ThirdPartyCall</td>
<td></td>
</tr>
<tr>
<td>makeCall</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_SERVICE_INITIATED</td>
</tr>
<tr>
<td>Parlay X 2.1 Part 3: Call Notification, interface CallNotification/SIP plug-in</td>
<td></td>
</tr>
<tr>
<td>notifyBusy</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_NETWORK_INITIATED</td>
</tr>
<tr>
<td>notifyNotReachable</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_NETWORK_INITIATED</td>
</tr>
<tr>
<td>notifyNoAnswer</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_NETWORK_INITIATED</td>
</tr>
<tr>
<td>notifyCalledNumber</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_NETWORK_INITIATED</td>
</tr>
<tr>
<td>Parlay X 2.1 Part 3: Call Notification, interface CallDirection/SIP plug-in</td>
<td></td>
</tr>
<tr>
<td>handleBusy</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_NETWORK_INITIATED</td>
</tr>
<tr>
<td>handleNotReachable</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_NETWORK_INITIATED</td>
</tr>
<tr>
<td>handleNoAnswer</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_NETWORK_INITIATED</td>
</tr>
</tbody>
</table>

Note: The method name is the name of an internal method corresponding to the message received from the network.
### Table 6-12 Transaction types for enhanced traffic paths

<table>
<thead>
<tr>
<th>Method</th>
<th>Transaction type</th>
</tr>
</thead>
<tbody>
<tr>
<td>handleCalledNumber</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_NETWORK_INITIATED</td>
</tr>
<tr>
<td></td>
<td>Parlay X 2.1 Part 3: Call Notification, interface CallNotification/OSA MPCC plug-in</td>
</tr>
<tr>
<td></td>
<td>Note: The method name is the name of an internal method corresponding to the message received from the network.</td>
</tr>
<tr>
<td>reportNotification</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_NETWORK_INITIATED</td>
</tr>
<tr>
<td></td>
<td>Parlay X 2.1 Part 4: Short messaging, interface SendSms</td>
</tr>
<tr>
<td>sendSms</td>
<td>TRANSACTION_TYPE_MESSAGING_SEND</td>
</tr>
<tr>
<td>sendSmsLogo</td>
<td>TRANSACTION_TYPE_MESSAGING_SEND</td>
</tr>
<tr>
<td>sendSmsRingtone</td>
<td>TRANSACTION_TYPE_MESSAGING_SEND</td>
</tr>
<tr>
<td></td>
<td>Parlay X 2.1 Part 4: Short messaging/SMPP plug-in</td>
</tr>
<tr>
<td></td>
<td>Note: The method name is the name of an internal method corresponding to the message received from the network.</td>
</tr>
<tr>
<td>receivedOriginatedSms</td>
<td>TRANSACTION_TYPE_MESSAGING_RECEIVE</td>
</tr>
<tr>
<td></td>
<td>Parlay X 2.1 Part 8: Terminal Status, interface TerminalStatus</td>
</tr>
<tr>
<td>getStatus</td>
<td>TRANSACTION_TYPE_USER_STATUS</td>
</tr>
<tr>
<td>getStatusForGroup</td>
<td>TRANSACTION_TYPE_USER_STATUS</td>
</tr>
<tr>
<td></td>
<td>Parlay X 2.1 Part 10: Call handling, interface CallHandling - OSA MPCC plug-in</td>
</tr>
<tr>
<td>reportNotification</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_NETWORK_INITIATED</td>
</tr>
<tr>
<td></td>
<td>Parlay X 2.1 Part 11: Audio call, interface AudioCall</td>
</tr>
<tr>
<td>playTextMessage</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_SERVICE_INITIATED</td>
</tr>
<tr>
<td>playAudioMessage</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_SERVICE_INITIITED</td>
</tr>
<tr>
<td>playVoiceXmlMessage</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_SERVICE_INITIITED</td>
</tr>
</tbody>
</table>
Transaction Types for Backwards Compatible Traffic Paths

Table 6-13 outlines the correlation between methods being invoked from either an application or the telecom network and the corresponding transaction type for backwards compatible traffic paths.

For backwards compatible traffic paths, the backwards compatible service capability supplies the statistics service with information on when to generate statistics.

<table>
<thead>
<tr>
<th>Method</th>
<th>Transaction type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parlay X 2.1 Part 2: Third Party Call, Interface: ThirdPartyCall</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_SERVICE_INITIATED.</td>
</tr>
<tr>
<td>makeCall</td>
<td>TRANSACTION_TYPE_CALL_CONTROL_SERVICE_INITIATED.</td>
</tr>
<tr>
<td>Parlay X 2.1 Part 4: Short messaging, interface SendSms</td>
<td>TRANSACTION_TYPE_MESSAGING_SEND</td>
</tr>
<tr>
<td>sendSms</td>
<td>TRANSACTION_TYPE_MESSAGING_SEND</td>
</tr>
<tr>
<td>sendSmsLogo</td>
<td>TRANSACTION_TYPE_MESSAGING_SEND</td>
</tr>
<tr>
<td>sendSmsRingtone</td>
<td>TRANSACTION_TYPE_MESSAGING_SEND</td>
</tr>
<tr>
<td>Parlay X 2.1 Part 4: Short messaging, network triggered SMS</td>
<td>TRANSACTION_TYPE_MESSAGING_RECEIVE</td>
</tr>
<tr>
<td>mmMessageArrived</td>
<td>TRANSACTION_TYPE_MESSAGING_MMS_RECEIVE</td>
</tr>
<tr>
<td>Parlay X 2.1 Part 5: Multimedia messaging, interface SendMessage</td>
<td>TRANSACTION_TYPE_MESSAGING_MMS_SEND</td>
</tr>
<tr>
<td>Parlay X 2.1 Part 5: Multimedia messaging, network triggered MMS</td>
<td>TRANSACTION_TYPE_MESSAGING_MMS_RECEIVE</td>
</tr>
</tbody>
</table>

Note: This is only a part of a traffic path, not including the backwards compatible network protocol plug-in.
### Table 6-13  Transaction types for backwards compatible traffic paths

<table>
<thead>
<tr>
<th>Method</th>
<th>Transaction type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parlay X 2.1 Part 6: Payment, interface AmountCharging</td>
<td></td>
</tr>
<tr>
<td>chargeAmount</td>
<td>TRANSACTION_TYPE_CHARGING_DIRECT</td>
</tr>
<tr>
<td>refundAmount</td>
<td>TRANSACTION_TYPE_CHARGING_DIRECT</td>
</tr>
<tr>
<td>Parlay X 2.1 Part 6: Payment, interface VolumeCharging</td>
<td></td>
</tr>
<tr>
<td>chargeVolume</td>
<td>TRANSACTION_TYPE_CHARGING_DIRECT</td>
</tr>
<tr>
<td>refundVolume</td>
<td>TRANSACTION_TYPE_CHARGING_DIRECT</td>
</tr>
<tr>
<td>Parlay X 2.1 Part 6: Payment, interface ReserveAmountCharging</td>
<td></td>
</tr>
<tr>
<td>reserveAmount</td>
<td>TRANSACTION_TYPE_CHARGING_RESERVE</td>
</tr>
<tr>
<td>reserveAdditionalAmount</td>
<td>TRANSACTION_TYPE_CHARGING_RESERVE</td>
</tr>
<tr>
<td>chargeReservation</td>
<td>TRANSACTION_TYPE_CHARGING_RESERVED</td>
</tr>
<tr>
<td>Parlay X 2.1 Part 6: Payment, interface ReserveVolumeCharging</td>
<td></td>
</tr>
<tr>
<td>reserveVolume</td>
<td>TRANSACTION_TYPE_CHARGING_RESERVE</td>
</tr>
<tr>
<td>reserveAdditionalVolume</td>
<td>TRANSACTION_TYPE_CHARGING_RESERVE</td>
</tr>
<tr>
<td>chargeReservation</td>
<td>TRANSACTION_TYPE_CHARGING_RESERVED</td>
</tr>
<tr>
<td>Parlay X 2.1 Part 9: Terminal Location, interface TerminalLocation</td>
<td></td>
</tr>
<tr>
<td>getLocation</td>
<td>TRANSACTION_TYPE_USER_LOCATION</td>
</tr>
<tr>
<td>getTerminalDistance</td>
<td>TRANSACTION_TYPE_USER_LOCATION</td>
</tr>
<tr>
<td>getLocationForGroup</td>
<td>TRANSACTION_TYPE_USER_LOCATION</td>
</tr>
<tr>
<td>Extended Web Services: WAP Push, interface PushMessage</td>
<td></td>
</tr>
<tr>
<td>sendPushMessage</td>
<td>TRANSACTION_TYPE_MESSAGE_SENDER_SEND</td>
</tr>
<tr>
<td>Method</td>
<td>Transaction type</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Extended Web Services: WAP Push, interface PushMessageNotification</td>
<td></td>
</tr>
<tr>
<td>resultNotificationMessage</td>
<td>TRANSACTION_TYPE_MESSAGE_SENDER_NOTIFY</td>
</tr>
</tbody>
</table>
Managing and Configuring the Global Counter Service

The following section describes how to configure and manage the Global Counter service. This service is deprecated in 3.0, as its functionality has been subsumed by the Budget Service. Nonetheless, the Global Counter service may be used by traffic paths or plug-in extensions originally created to be used with Network Gatekeeper 2.2. It is preserved in order to ensure backwards compatibility.

- Configuration and Management
- Reference: attributes and operations for SLEE_global_counter

**Configuration and Management**

All configuration and management is performed in SLEE_global_counter.

**Configuration workflow**

1. Decides if volatile counters should be cleaned up automatically, use Attribute: CleanupOfVolatileEnabled.
2. Defines synchronization intervals for counters. Use:
   - Operation: setCounterSynchronizationInfo
   - Operation: setSubCounterSynchronizationInfo

**Management operations**

Management of counters:
Managing and Configuring the Global Counter Service

Operation: explicitlyCleanupVolatile
Operation: getSubCounters
Operation: resetGlobalCounter
Operation: resetSubCounter

Reference: attributes and operations for SLEE_global_counter

Below is a list of attributes and operations for configuration and maintenance.

- Attribute: CounterSynchronizationInfo (r)
- Attribute: CleanupOfVolatileEnabled
- Attribute: GlobalCounters (r)
- Operation: explicitlyCleanupVolatile
- Operation: getSubCounterSynchronizationInfo
- Operation: getSubCounters
- Operation: resetGlobalCounter
- Operation: resetSubCounter
- Operation: setCounterSynchronizationInfo
- Operation: setSubCounterSynchronizationInfo

Attribute: CounterSynchronizationInfo (r)
Scope: Server
Displays counter synchronization information.

Attribute: CleanupOfVolatileEnabled
Scope: Server
Format: boolean
Specifies if the cleanup of volatile counters is enabled or not.

**Attribute: GlobalCounters (r)**
Scope: Server
Displays a list of global counters.

**Operation: explicitlyCleanupVolatile**
Scope: Server
Cleans up dead volatile counters. Returns information on the number of counters cleaned up.
Signature:
explicitlyCleanupVolatile()

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: getSubCounterSynchronizationInfo**
Scope: Cluster
Displays information about synchronization and persistence for a given global counter.
Signature:
getSubCounterSynchronizationInfo(counterId: String, subCounterId: String)
Table 7-2 getSubCounterSynchronizationInfo

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>counterId</td>
<td>ID of global counter.</td>
</tr>
<tr>
<td>subCounterId</td>
<td>ID of subcounter.</td>
</tr>
</tbody>
</table>

**Operation: getSubCounters**

Scope: Cluster

Displays a list of all subcounters for a global counter.

Signature:

getSubCounters(globalCounterID: String)

Table 7-3 getSubCounters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>globalCounterID</td>
<td>ID of global counter.</td>
</tr>
</tbody>
</table>

**Operation: resetGlobalCounter**

Scope: Cluster

Resets a global counter. Subcounters are not reset.

Signature:

resetGlobalCounter(globalCounterID: String)
Operation: resetSubCounter
Scope: Cluster
Resets a subcounter for global counter.
Signature:
resetGlobalCounter(globalCounterID: String)

Table 7-5  resetSubCounter
resetSubCounter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>globalCounterID</td>
<td>ID of global counter.</td>
</tr>
<tr>
<td>subCounterId</td>
<td>ID of subcounter.</td>
</tr>
</tbody>
</table>

Operation: setCounterSynchronizationInfo
Scope: Server
Defines criteria for deciding when counters should be synchronized between Network Gatekeeper servers
Signature:
setCounterSynchronizationInfo(config.counterBroadcastTimeIntervalMillis: int, config.counterBroadcastIncrements: int)
Managing and Configuring the Global Counter Service

Operation: `setSubCounterSynchronizationInfo`

Scope: Cluster

Defines criteria for when subcounters should be synchronized between Network Gatekeeper servers.

Signature:

```java
setSubCounterSynchronizationInfo(config.counterBroadcastTimeIntervalMillis: int, config.counterBroadcastIncrements: int)
```

Table 7-6 `setCounterSynchronizationInfo`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>config.counterBroadcastTimeIntervalMillis</td>
<td>Time interval between synchronizations. Given in milliseconds.</td>
</tr>
<tr>
<td>config.counterBroadcastIncrements</td>
<td>not used.</td>
</tr>
</tbody>
</table>

Table 7-7 `setCounterSynchronizationInfo`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>config.counterId</td>
<td>ID of global counter.</td>
</tr>
<tr>
<td>config.subCounterId</td>
<td>ID of subcounter.</td>
</tr>
<tr>
<td>config.counterPersistIncrementsPercentage</td>
<td>Maximum number of increments before the subcounter information is stored in persistent storage. This setting is per persistent Global counter. The number of increments is given in percentage of the number of transactions given in the tag <code>&lt;quota&gt;</code> in the SLA.</td>
</tr>
</tbody>
</table>
config.counterBroadcastIncrementsPercentage

Maximum number of increments before a broadcast of the subcounter information in one server to subcounters in other servers. This setting is per persistent counter. Given in percentage of the number of transactions given in the tag <quota> tag in the SLA.

config.counterPersistentIncrementsInterval

Maximum time interval before the subcounter information is stored in persistent storage. This setting is per persistent counter. Given in milliseconds.

config.counterBroadcastIncrementsInterval

Maximum number of increments before a broadcast of the subcounter information to other Network Gatekeeper servers. This setting is per persistent counter. The number of increments is given in percentage of the number of transactions given in the tag <quota> in the SLA.
Managing and Configuring the SCS Manager

The following section describes how to configure and manage the SCS Manager service. This service is deprecated. It is used by the backwards compatible Messaging Traffic paths, and possibly by traffic path or plug-in extensions created in the context of Network Gatekeeper 2.2. It is preserved in order to ensure backwards compatibility. This functionality is not applicable for enhanced traffic paths.

- Configuration and Management
- Reference: attributes and operations for SCS_manager

Introduction

The SCS manager is responsible for routing network-initiated requests from a plug-in to an appropriate backwards compatible ESPA Service Capability. The routing is performed using criteria which define which ESPA Service Capability the route requires by using its SCS type, sub-type, and address criteria that are matched with the destination address provided in the request between the plug-in and the service capability module.

Configuration and Management

All configuration and management is performed in SCS_manager.

Configuration workflow

1. Define a criterion, use Operation: addCriteria.
Management operations
Management of counters:

- Operation: addCriteria
- Operation: listCriteria
- Operation: removeCriteria

Reference: attributes and operations for SCS_manager
Below is a list of attributes and operations for configuration and maintenance.

- Attribute: SCSList (r)
- Attribute: SCSMgrIOR (r)
- Attribute: TypeList (r)
- Operation: addCriteria
- Operation: listCriteriaHandlers
- Operation: listCriteria
- Operation: registerSCS
- Operation: registerSCSFromURL
- Operation: removeCriteria
- Operation: saveSCSMgrIOR
- Operation: unregisterSCS
- Operation: unregisterSCSFromURL

Attribute: SCSList (r)
Scope: Cluster
Displays information about registered backwards compatible ESPA service capability modules.
**Attribute: SCSMgrIOR (r)**
Scope: Server
Displays information about the CORBA IOR for the SCS manager service.

**Attribute: TypeList (r)**
Scope: Cluster
Displays information about types registered in the SCS manager.

**Operation: addCriteria**
Scope: Server
Creates selection criteria. See Introduction.

Signature:

```
addCriteria(scsType: String, scsProperties: String, criteria: String)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scsType</td>
<td>Specifies to which type of service capability an incoming message shall be routed. The routing is based on the destination address of the incoming message. See Attribute: SCSList (r)</td>
</tr>
<tr>
<td></td>
<td>Criteria definition is only needed for service capabilities of type MESSAGING_TYPE, and possible by traffic path extensions built for Network Gatekeeper 2.2.</td>
</tr>
<tr>
<td>scsProperties</td>
<td>Enter</td>
</tr>
<tr>
<td></td>
<td>• SUBTYPE GMS for ESPA_messaging</td>
</tr>
<tr>
<td>criteria</td>
<td>A regular expression for the destination address.</td>
</tr>
<tr>
<td></td>
<td>For example ^[0-5].* matching all destination addresses starting with 0, 1, 2, 3, 4 or 5.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The destination address expressions must match the addresses that are used for mailboxes in ESPA_messaging.</td>
</tr>
</tbody>
</table>
**Operation: listCriteriaHandlers**

Scope: Cluster

Lists the SCS types that have an associated CriteriaHandler.

Signature:

```java
listCriteriaHandlers()
```

**Table 8-2 listCriteriaHandlers**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: listCriterias**

Scope: Cluster

Displays information about registered criteria.

Signature:

```java
listCriterias(scsType: String, startIndex: int, maxCount: int)
```

**Table 8-3 listCriterias**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scsType</td>
<td>SCS type.</td>
</tr>
</tbody>
</table>
Operation: registerSCS
Not applicable.

Operation: registerSCSFromURL
Not applicable.

Operation: removeCriteria
Scope: Cluster
Removes a criterion.
Signature:
removeCriteria(scsType: String, id: int)

Table 8-4 removeCriteria

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scsType</td>
<td>SCS type.</td>
</tr>
<tr>
<td>id</td>
<td>ID of criterion. This ID was created when the criterion was added. See Attribute: SCSList (r)</td>
</tr>
</tbody>
</table>
Operation: saveSCSMgrIOR
Not applicable.

Operation: unregisterSCS
Not applicable.

Operation: unregisterSCSFromURL
Not applicable.
Managing and Configuring the SLEE Event Service

The following section describes how to configure and manage the SLEE_event service. This service is deprecated in 3.0. It is used internally in Network Gatekeeper, and possibly by external event listeners. It is preserved in order to ensure backwards compatibility. All new events are managed by the EDRService.

- Configuration and Management
- Reference: attributes and operations for SLEE_event

Configuration and Management

All configuration and management is performed in SLEE_event.

Configuration workflow

1. Decide if events distributed via SLEE_event service should be persisted to database or not - use Attribute: DbEnabled.
2. Decide which events should be written to the database - use Operation: setFilterLevel.

Management operations

Management for external listeners:

- Operation: isListenerRegistered
- Operation: removeEventListener
• To view the CORBA IOR for SLEE_event service itself, use Attribute: EventLoggerIOR (r)

Management of event data:

• Operation: copyEventDataToFile
• Operation: deleteEvent
• Operation: deleteEvents
• Operation: dumpEventData
• Operation: getEvent
• Operation: listEvents

Reference: attributes and operations for SLEE_event

Below is a list of attributes and operations for configuration and maintenance.

• Attribute: FilterLevel (r)
• Attribute: DbEnabled
• Attribute: EventLoggerIOR (r)
• Operation: addEventListener
• Operation: copyEventDataToFile
• Operation: deleteEvent
• Operation: deleteEvents
• Operation: dumpEventData
• Operation: getEvent
• Operation: isListenerRegistered
• Operation: listEvents
• Operation: removeEventListener
• Operation: setFilterLevel
**Attribute: FilterLevel (r)**
Read-only.
Scope: Server
Format: int
Specifies which events are written to the event log. The active filter level is represented with a digit according the table below:
- 1 - all events are logged
- 2 - medium and high importance are logged
- 3 - only high importance events are logged.

**Attribute: DbEnabled**
Scope: Cluster
Format: boolean
Defines if events shall be written to the database or not.

**Attribute: EventLoggerIOR (r)**
Scope: Cluster
Format: String
String representation of event logger CORBA IOR.

**Operation: addEventListener**
Scope: Cluster/Server
Registers an event listener.
Signature:
```
addEventListener(listenerIor : String, registerInAllInstances :boolean)
```
**Operation: copyEventDataToFile**

Scope: Cluster

Copies event data from the database to file.

Signature:

```java
copyEventDataToFile(fileName: String, fromDate: String, toDate: String)
```

**Table 9-2 copyEventDataToFile**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>The filename for the report. Must include an absolute path. The file is created on the local file system of the selected server.</td>
</tr>
<tr>
<td>fromDate</td>
<td>Specifies the start date and time for the time interval. Format is YYYY-MM-DD hh:mm:ss.</td>
</tr>
<tr>
<td>toDate</td>
<td>Specifies the end date and time for the time interval. Format is YYYY-MM-DD hh:mm:ss.</td>
</tr>
</tbody>
</table>
Operation: deleteEvent

Scope: Cluster

Deletes an event with a certain ID from the database.

Signature:

```java
deleteEvent(eventId: String)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventId</td>
<td>ID of the event.</td>
</tr>
</tbody>
</table>

Operation: deleteEvents

Scope: Cluster

Deletes a range of events matching the given criteria from the database.

Signature:

```java
deleteEvents(serviceName: String, level: int, fromDate: String, toDate: String, identifier: int)
```
Operation: dumpEventData

Scope: Cluster

Dumps event data to file.

Note: Dumping the data from the database removes that data from the database. Enough disk space must be available to write the data. Each entry in the charging data table may take up to 300 bytes when written to file.

Signature:

Table 9-4 deleteEvents

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>Name of service from which the event originated. &lt;br&gt;&lt;br&gt;Note: Leave empty to match all service names.</td>
</tr>
<tr>
<td>level</td>
<td>Level of event, use:&lt;br&gt;• 1 for LOW&lt;br&gt;• 2 for MEDIUM&lt;br&gt;• 3 for HIGH&lt;br&gt;Use level 0 to match all event levels.</td>
</tr>
<tr>
<td>fromDate</td>
<td>Specifies the start date and time for the time interval. Format is YYYY-MM-DD hh:mm:ss. &lt;br&gt;&lt;br&gt;Note: Leave empty for all up to the date and time specified in toDate.</td>
</tr>
<tr>
<td>toDate</td>
<td>Specifies the end date and time for the time interval. Format is YYYY-MM-DD hh:mm:ss. &lt;br&gt;&lt;br&gt;Note: Leave empty for all from the date and time specified in fromDate until current date and time.</td>
</tr>
<tr>
<td>identifier</td>
<td>Identifier of event. Use identifier 0 to match all events.</td>
</tr>
</tbody>
</table>
dumpEventData(fileName)

Table 9-5 dumpEventData

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>The filename. Must include an absolute path. The file is created on the local file system of the selected server.</td>
</tr>
</tbody>
</table>

**Operation: getEvent**

Scope: Cluster
Displays information about certain event.
Signature:

getEvent(eventId: long)

Table 9-6 eventId

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventId</td>
<td>Id of the event.</td>
</tr>
</tbody>
</table>

**Operation: isListenerRegistered**

Scope: Cluster
Displays whether a listener IOR is registered or not.
Signature:

isListenerRegistered(listenerIOR: String)
### Table 9-7 isListenerRegistered

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listenerIOR</td>
<td>CORBA IOR of the listener.</td>
</tr>
</tbody>
</table>

### Operation: listEvents

**Scope:** Cluster

Displays a range of events matching the given criteria from the database.

**Signature:**

```java
listEvents(serviceName: String, level: int, fromDate: String, toDate: String, identifier: int)
```

### Table 9-8 listEvents deleteEvents

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>Name of service from which the event originated.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Leave empty to match all service names.</td>
</tr>
<tr>
<td>level</td>
<td>Level of event, use:</td>
</tr>
<tr>
<td></td>
<td>• 1 for LOW</td>
</tr>
<tr>
<td></td>
<td>• 2 for MEDIUM</td>
</tr>
<tr>
<td></td>
<td>• 3 for HIGH</td>
</tr>
<tr>
<td></td>
<td>Use level 0 to match all event levels.</td>
</tr>
<tr>
<td>fromDate</td>
<td>Specifies the start date and time for the time interval.</td>
</tr>
<tr>
<td></td>
<td>Format is YYYY-MM-DD hh:mm:ss.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Leave empty for all up to the date and time specified in toDate.</td>
</tr>
</tbody>
</table>
Table 9-8  listEvents

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| toDate | Specifies the end date and time for the time interval. Format is YYYY-MM-DD hh:mm:ss.  
  Note: Leave empty for all from the date and time specified in fromDate until current date and time. |
| identifier | Identifier of event. Use identifier 0 to match all events. |

Operation: removeEventListener

Scope: Cluster/Server

Unregisters an event listener.

Signature:

removeEventListener(listenerIOR: String, removeFromAllInstances: boolean)

Table 9-9  removeEventListener

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listenerIOR</td>
<td>CORBA IOR to the event listener.</td>
</tr>
<tr>
<td>removeFromAllInstances</td>
<td>If the listener should be unregistered from all servers, enter true. If the listener should be unregistered only in the current server, enter false.</td>
</tr>
</tbody>
</table>

Operation: setFilterLevel

Scope: Server

Format: int

Specifies which events are written to the event log.
Managing and Configuring the SLEE Event Service

Signature:

```
setFilterLevel(level: int)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>level</td>
<td>The active filter level is represented with a digit according to:</td>
</tr>
<tr>
<td></td>
<td>1 - all events are logged</td>
</tr>
<tr>
<td></td>
<td>2 - medium and high importance are logged</td>
</tr>
<tr>
<td></td>
<td>3 - only high importance events are logged.</td>
</tr>
</tbody>
</table>
Setting Up Geographic Redundancy

The following section describes how to setup geographically redundant site pairs and the maintenance attributes and operations for the geographic redundancy service once it is set up. It also provides a workflow for the configuration:

- Introduction
- Configuration Workflow
- Reference: attributes and operations for GeoRedundantService

Introduction

The geographic redundancy service is responsible for replicating data, so that applications can switch from one site to another and have the data necessary for SLA enforcement available at the second, remote, site. For more information about geographic redundancy, refer to WebLogic Network Gatekeeper Architectural Overview.

Each geographic site has a name. This name is used for looking up data relevant to the site pair. All configuration is done locally for the site, so the remote site must be defined in the local site.

Configuration Workflow

Below is an outline for configuring the geographic redundant service. This must be done at both sites.

1. Define the ID of the local site in Attribute: GeoSiteId.
2. Define when alarms should be raised due to access problems to the remote site in Attribute: RemoteSiteReachabilityAlarmThreshold.

3. Define the remote site in Operation: setSiteAddress.

Reference: attributes and operations for GeoRedundantService

Below is a list of attributes and operations for configuration and maintenance.

- **Attribute: RemoteSiteReachabilityAlarmThreshold**
- **Attribute: GeoSiteId**
- **Operation: getSiteAddress**
- **Operation: listRemoteSites**
- **Operation: removeSite**
- **Operation: setSiteAddress**

**Attribute: RemoteSiteReachabilityAlarmThreshold**

Scope: Cluster
Format: int
Specifies the number of attempts made by a site to reach peer site before raising alarm.
Whenever the peer sites fail to establish a connection the number of times defined in RemoteSiteReachabilityAlarmThreshold, a connection lost alarm is raised.

**Attribute: GeoSiteId**

Scope: Cluster
Format: String
Defines the name of this geographic redundant site. This name is used as key for all operations on the site - see Operation: setSiteAddress, Operation: getSiteAddress, Operation: removeSite.

**Operation: getSiteAddress**

Scope: Cluster
Reference: attributes and operations for GeoRedundantService

Signature:

getSiteAddress(Site name: String)

Displays the address of a given site.

Table 10-1  getSiteAddress

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site name</td>
<td>The name of the site.</td>
</tr>
</tbody>
</table>

**Operation: listRemoteSites**

Scope: Cluster

Signature:

listRemoteSites()

Displays a list of registered remote sites.

Table 10-2  listRemoteSites

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Operation: removeSite**

Scope: Cluster

Signature:

removeSite(Site name: String)

Removes a site definition.
Setting Up Geographic Redundancy

**Table 10-3 removeSite**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site name</td>
<td>The site name.</td>
</tr>
</tbody>
</table>

**Operation: setSiteAddress**

Scope: Cluster

Specifies the address of a remote site.

Signature:

\[
\text{setSiteAddress}(\text{Site name}: \text{String}, \text{Address}: \text{String})
\]

**Table 10-4 setSiteAddress**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site name</td>
<td>Name of the remote site.</td>
</tr>
<tr>
<td>Address</td>
<td>JNDI URL of the network tier for the remote site, according to WebLogic Server addressing standards:</td>
</tr>
<tr>
<td></td>
<td>(&lt;\text{protocol}://&lt;\text{host}&gt;:&lt;\text{port}&gt;)</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>t3://host1:port,host2:port</td>
</tr>
</tbody>
</table>
Setting up WS-Policy and JMX Policy

The following section describes enabling security settings for Web Services and for OAM MBeans.

Introduction

One of the first things you must do in setting up your Network Gatekeeper installation is to make decisions about two key forms of security for your installation: Web Services security and MBean security. Web Services Security controls Network Gatekeeper’s interactions with Application Service Providers. MBean security controls who can have access to the Runtime MBean Server within your WLS installation, the mechanism that allows OAM procedures to be done.

- **Web Services Security**
- **JMX Policy**

Web Services Security

Web Services Security provides end-to-end message-level security for web services through an implementation of the WS-Security standard.

WS-Security defines a mechanism for adding three levels of security to SOAP messages:

- Authentication tokens. WS-Security authentication tokens lets an application provide a user name and password or X509 certificate for the purpose of authentication headers. With additional setup, SAML can also be used for authentication.
Setting up WS-Policy and JMX Policy

Note: Out of the box, Network Gatekeeper is pre-configured to use user name/password authentication.

- **XML encryption.** WS-Security’s use of W3C’s XML encryption standard enables the XML body or portion of it to be encrypted to ensure message confidentiality.

- **XML digital signatures.** WS-Security’s use of W3C’s XML digital signatures lets the message be digitally signed to ensure message integrity. The signature is based on the content of the message itself (by applying the hash function and public key), so if the message is altered en route, the signature becomes invalid.

Network Gatekeeper uses WebLogic Server mechanisms for Web Services security—see:

- *Programming Web Services for WebLogic Server,*
  http://e-docs.bea.com/wls/docs92/webserv/security.html

- *Understanding WebLogic Security,*
  http://edocs.beasys.com/wls/docs92/secintro/concepts.html

- *Web Services Security specifications,*

Authentication is handled transparently by WS-Security and subsequently by the configured authentication providers and login modules of the WebLogic Security framework. WS-Security also supports signing and encrypting a message by providing a security token hierarchy associated with the keys used for signing and encryption (for message integrity and confidentiality).

The following steps outline the general WebLogic security configurations that can be performed, either automatically using a script or manually from the Management Console.

- For SAML tokens only: Configure WebLogic SAML Identity Assertion Provider which authenticates users based on SAML assertions and SAML credential mapping provider. The SAML Identity Assertion Provider is required only if you are using SAML assertions.

- Configure Policies for WS-Security as described below.

**Configuration workflow: WS-Policy**

This section outlines how to apply an existing WS-Policy and where to find more information on creating and using custom WS-Policies.
Apply WS-Policy to a Web Service: Quick start

This section outlines how to apply a WSSE policy to a Web Service endpoint in Network Gatekeeper.

Note: Out of the box, Network Gatekeeper is pre-configured to use the auth.xml policy file, which requires the use of username/password authentication. It is also set up to require this authentication only for inbound traffic. The following description is provided in case this particular mechanism does not cover the needs of your installation.

Standard WebLogic Server mechanisms are used - see http://e-docs.bea.com/wls/docs92/ConsoleHelp/taskhelp/webservices/ConfigureWSPolicyFile.html for a full description.

Starting in WebLogic Console:

1. In the **Domain Structure** pane, select **Deployments**.

2. In **Summary of Deployments** page, expand **wlng_at**.

3. Click on a Web Service to apply Web Services security for, for example **AudioCallService**. All Web Services are named according to the interface they implement. This shows the page **Settings for <Web Service>**

4. Click the **Configuration** tab.

5. Click **WS-Policy** sub-tab.

6. Click **Service endpoint <Web Service>**.

7. Choose which security policy to apply to the endpoint:

   a. Select the appropriate ws-policy file in **Available Endpoint Policies**, see **Available default WS-Policies**.

   b. Move it to the list in **Chosen Endpoint Policies** by clicking on the arrow button.

   c. When the WS-Policy files have been chosen, click **OK**.

8. In the **Save Deployment Plan Assistant** you have the choice of where to store the deployment plan. You should choose `<beahome>/domains/<wlng-domain>./servers/WLNG_AT1/stage/wlng_at/plan/Plan.xml` (For a single instance development machine, substitute your server name for WLNG_AT1)

9. Apply the changes.
Note: Applying a security policy to a Web Service establishes, by default, both inbound and outbound security policies. Because there is no way for Network Gatekeeper to know what security policies may be required by a client to which it is returning a notification, outbound security must be turned off. If you wish to secure the link by which Network Gatekeeper returns notifications, you should use SSL.

To turn off outbound security associated with a particular WS-Policy file, you must edit the plan.xml file that is created when you attach Policy to a Web Service, as in step 8 above. The default location is:

`/domains/<wlng-access-network-domain>/servers/WLNG_AT1/stage/wlng_at/plan/Plan.xml`, but your location may vary. Make sure the `<value>` element is set to `inbound` as in the following stanza:

Listing 11-1  Plan.xml snippet to be edited

```
<variable>
   <name>WsPolicy_policy:Auth.xml_Direction_11745107731400</name>
   <value>inbound</value>
</variable>
```

Remove WS-Policy from a Web Service

Network Gatekeeper 2.2 used a slightly different mode of authentication than the WS-Security model of version 3.0. Network Gatekeeper can be configured to support applications designed to work using the older model, but the WS-Policy that is configured out of the box must be removed.

Note: The easiest way to do this is to make these changes before you start Network Gatekeeper up the first time. Certain configuration values are cached on start up. So, for example, if you started Network Gatekeeper during the Post-Install phase in order to set up additional JMS Servers, you will need to redeploy the `wlng_at.ear` file after you have made your changes.

To remove the policy files from a Web Service:

1. Unjar the `wlng_at.ear` file.
2. Unjar the war file for the Web Service in question.
3. Modify the weblogic-webservices-policy.xml file for that Web Service to remove the policy entries.

   Note: See Listing 11-2 and Listing 11-3 for before and after snippets.

Listing 11-2   weblogic-webservices-policy.xml with policy entries

```xml
<?xml version='1.0' encoding='UTF-8'?>
<webservice-policy-ref xmlns="http://www.bea.com/ns/weblogic/90"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <port-policy>
    <port-name>AudioCall</port-name>
    <ws-policy>
      <uri>policy:Auth.xml</uri>
      <direction>inbound</direction>
    </ws-policy>
  </port-policy>
</webservice-policy-ref>
```

Listing 11-3   weblogic-webservices-policy.xml with policy entries removed

```xml
<?xml version='1.0' encoding='UTF-8'?>
<webservice-policy-ref xmlns="http://www.bea.com/ns/weblogic/90"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"></webservice-policy-ref>
```

4. Re-package the war file with the modified weblogic-webservices-policy.xml file

5. Re-package the wlng_at.ear file

6. Copy the modified ear file to the domain directory of the Admin Server. In the default installation, this would be <bea_home>/user_projects/domains/<wlng-domain>/.
7. If you have previously run Network Gatekeeper, you should now re-deploy the `wlng_at.ear` file using the Management Console.

**Create and use custom a custom WS-Policy**


**Available default WS-Policies**

WS-Policy files can be used to require applications clients to authenticate, digitally encrypt, or digitally sign SOAP messages. Out-of-the-box Network Gatekeeper supplies files to do those three things, respectively: auth.xml, encrypt.xml, and sign.xml. If the built-in WS-Policy files do not meet your security needs, you can build custom policies.

See [http://e-docs.bea.com/wls/docs92/webserv/security.html](http://e-docs.bea.com/wls/docs92/webserv/security.html) for a description.

WS-Policy assertions are used to specify a Web Services’ requirements for digital signatures and encryption, along with the security algorithms and authentication mechanisms that it requires, for example Policy for SAML.

**JMX Policy**

Access to the OAM functionality of WebLogic Network Gatekeeper - both through the Console and through external mechanisms - is made using Java Management Extension (JMX) MBeans. Access to these MBeans is controlled by JMX Policy, which associates administrative user groups with access privilege levels. When Network Gatekeeper is installed, there are no controls established by default on access to the OAM MBeans. Each installation must make decisions about access based on its own needs.
Administrative Groups

There are four administrative user privilege levels, which are assigned to four Network Gatekeeper Administrative Groups:

<table>
<thead>
<tr>
<th>User Privilege Level</th>
<th>Network Gatekeeper Administrative Group Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anonymous</td>
<td>Everyone</td>
</tr>
<tr>
<td>Read only</td>
<td>WLNG_ReadOnly</td>
</tr>
<tr>
<td>Standard read and write</td>
<td>WLNG_ReadWrite</td>
</tr>
<tr>
<td>Administrator</td>
<td>WLNG_Administrators</td>
</tr>
</tbody>
</table>

Administrative users and groups are set up as described in Managing Network Gatekeeper Administrative Users. To control how these users have access to MBeans, and thus OAM functionality, you must assign JMX Policy to these user groups. You use the WLS Console to do this, as described in Create JMX Policies. Each policy can do the following:

- Control read access for all an MBean’s attributes or for specific attributes that you select.
- Control write access for all an MBean’s attributes or for specific attributes that you select.
- Control invoke access for all an MBean’s operations or for specific operations that you select

For example, to give a user complete access to an MBean, select WLNG_Administrator’s Group in the policy condition.

Administrative Service Groups

In addition to controlling access to OAM functionality in a general way - ReadOnly, ReadWrite, etc. - you may also wish to control access by Service group. So, for example, if you have users whose job is limited to setting up and managing Application Service Providers through a system using the Partner Relationship Management interfaces, you might want to give them, and only them, ReadWrite privileges, but only to a subset of the available MBeans, those having to do with the operator part of those transactions. To do this you have to create custom XACML policies to attach to these subsets. See Using XACML Documents to Secure WebLogic Resources for a detailed description of this process. The basic process includes:
Setting up WS-Policy and JMX Policy

- Determine the special identifier (called the resourceId) for each MBean
- Create an XACML policy for a security role
- Specify one or more Rule elements that define which users, groups, or roles belong to the new security role
- Attach this role to the MBean by way of the resourceId.
Managing and Configuring the SNMP service

The following section describes how to configure and manage the SNMP service.

- Configuration and management
- Reference: attributes and operations for SLEE_snmp

**Introduction**

The SNMP service is responsible for collecting alarms and distributing them as SNMP traps.
The SNMP service acts as an internal alarm listener and sends traps (or notifications) to any registered SNMP trap listener. There is a 1:1 relationship between alarms and SNMP traps. The MIB file defining the SNMP traps is based on the content of the alarm.xml file.

Each individual alarm ID is used to generate the Object Identifier for each SNMP trap.

The SNMP traps sent are constructed by:

\[
\text{IANA-registered private enterprise ID} + \text{BEA ID} + \text{Network Gatekeeper ID} + \text{"0"} + \text{alarm identifier}
\]

- The IANA-registered enterprise ID is configurable using Attribute: EnterpriseObjectIdentifier.
- The BEA ID is static, and the value is 140
- The Network Gatekeeper ID is static, and the value is 627
- The alarm identifier is defined in alarm.xml, in the attribute \text{id} of the \text{<alarm>} element.

The MIB file BEA-WLNG-MIB is located in the sub-directory snmp in the domain directory.
Configuration and management

All configuration and management is performed in the managed object SLEE_snmp.

Configuration workflow

1. Define the SNMP community string in Attribute: Community.
2. Define the Enterprise Object Identifier in Attribute: EnterpriseObjectIdentifier. This attribute has a default value and should normally not be changed.
3. Define the SNMP version to use, in Operation: setSNMPVersion.
4. Define which alarm severity levels that shall be distributed as SNMP traps, in Operation: setSeverityFilter.

Management operations

The following management operation are used to manage trap receivers:

- Operation: addTrapReceiver
- Operation: deleteTrapReceiver
- Operation: listTrapReceivers

Reference: attributes and operations for SLEE_snmp

Below is a list of attributes and operations for configuration and maintenance.

- Attribute: Community
- Attribute: EnterpriseObjectIdentifier
- Attribute: RepeatedTraps (r)
- Attribute: SNMPVersion (r)
- Attribute: SeverityFilter (r)
- Operation: addTrapReceiver
- Operation: deleteTrapReceiver
Managing and Configuring the SNMP service

- **Operation**: listTrapReceivers
- **Operation**: setRepeatedTraps
- **Operation**: setSNMPVersion
- **Operation**: setSeverityFilter

**Attribute: Community**
Scope: Server
Specifies the SNMP community address.
Default value is private.

**Attribute: EnterpriseObjectIdentifier**
Scope: Server
Specifies the base enterprise object identifier used for the SNMP traps.
BEA ID and Network Gatekeeper IDs are fixed and appended to this ID.
For each individual SNMP trap, the alarm ID for the alarm is appended per alarm.

**Attribute: RepeatedTraps (r)**
Scope: Cluster
Displays information about the number of times each SNMP trap is sent to each configured manager.

**Attribute: SNMPVersion (r)**
Scope: Cluster
Displays information about which SNMP version is used.
If 0, SNMP v1 is used.
If 1, SNMP v2 is used.

**Attribute: SeverityFilter (r)**
Scope: Cluster
Displays information about current severity filter setting. Only alarms with a severity that exceeds or is equal to the specified severity filter will cause a trap to be generated:

- CRITICAL = 4
- MAJOR = 3
- MINOR = 2
- WARNING = 1

**Operation: addTrapReceiver**

Scope: Cluster

Adds a receiver for the SNMP traps.

Signature:

```java
addTrapReceiver(address: String, port: int)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>IP-address for the trap receiver.</td>
</tr>
<tr>
<td>port</td>
<td>Port for the trap receiver.</td>
</tr>
</tbody>
</table>

**Operation: deleteTrapReceiver**

Scope: Cluster

Deletes a previously added trap receiver.

Signature:

```java
deleteTrapReceiver(id: int)
```
Table 12-2  deleteTrapReceiver

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>ID of the trap receiver to delete. Use Operation: listTrapReceivers to list the IDs.</td>
</tr>
</tbody>
</table>

**Operation: listTrapReceivers**

Scope: Cluster

Displays a list of all registered trap receivers.

Signature:

`listTrapReceivers()`

Table 12-3  listTrapReceivers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: setRepeatedTraps**

Scope: Cluster

Specifies how many times each trap is sent to a configured manager.

Signature:

`setRepeatedTraps(NrOfRepeatedTraps: int)`
Table 12-4 setRepeatedTraps

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NrOfRepeatedTraps</td>
<td>The number of times each trap shall be sent.</td>
</tr>
</tbody>
</table>

Operation: setSNMPVersion

Scope: Cluster

Specifies the SNMP version to use.

Signature:

```
setSNMPVersion(version: int)
```

Table 12-5 setRepeatedTraps

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>ID for the SNMP version to use. Enter:</td>
</tr>
<tr>
<td></td>
<td>• 0 for SNMP v.1</td>
</tr>
<tr>
<td></td>
<td>• 1 for SNMP v.2</td>
</tr>
</tbody>
</table>

Operation: setSeverityFilter

Scope: Cluster

Specifies a filter to define which alarms should be sent as SNMP traps.

Signature:

```
setSeverityFilter(severity: int)
```
Table 12-6  setSeverityFilter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>severity</td>
<td>The filter level is represented with a digit. The filter level is specified using an identifier.</td>
</tr>
<tr>
<td></td>
<td>• 1 - all alarms are sent as traps.</td>
</tr>
<tr>
<td></td>
<td>• 2 - minor, major and critical alarms are sent as traps.</td>
</tr>
<tr>
<td></td>
<td>• 3 - major and critical alarms are sent as traps.</td>
</tr>
<tr>
<td></td>
<td>• 4 - only critical alarms are sent as traps.</td>
</tr>
</tbody>
</table>
CHAPTER 13

Configuring WebLogic SIP Server Integration

The following section describes how to configure the WebLogic Network Gatekeeper and WebLogic SIP Server integration. A prerequisite is that the Network Gatekeeper and WebLogic SIP Server integration modules both be installed in your WebLogic SIP server.

Note: See the “Completing Post-Installation” chapter in the Installation Guide for more information on these integration modules.

- Introduction
- Configuration Workflow
- Reference: attributes and operations

Introduction

The WebLogic Network Gatekeeper and WebLogic SIP Server integration must be configured in WebLogic SIP Server so that requests can be correlated between WebLogic SIP Server and Network Gatekeeper. The WebLogic SIP Server part of the configuration is described in this section. The Network Gatekeeper configuration is described in the description of the relevant traffic paths.

Configuration Workflow

Configure the WebLogic SIP Server part of the Network Gatekeeper and WebLogic SIP Server integration:
1. In the Administration Console for WebLogic SIP Server, click on **WLNG Integration** in the **Domain Structure** group.

   This opens the **WLNG Integration Configuration** page.

2. Click **Lock & Edit**.

3. In the General Tab, enter data for:
   - **Attribute**: Username
   - **Attribute**: Password

4. Click **Save**.

5. In the ThirdParty Call Control tab, enter data for:
   - **Attribute**: Controller URI

6. In the Presence tab, enter data for:
   - **Attribute**: Suggested subscription lifetime
   - **Attribute**: Presence Server address

7. Click **Activate Changes**.

The following attributes should not be changed:

- In Third Party Call Control tab:
  - Call Manager Class Name
  - SIP Servlet Name

- In Call Notification tab:
  - Call Notification Plugin JNDI Name:
  - Call Direction Plugin JNDI Name:

**Reference: attributes and operations**

Below is a list of attributes and operations for configuration and maintenance.

- **Attribute**: WLNG JNDI URL
- **Attribute**: Username
- **Attribute**: Password
**Attribute: WLNG JNDI URL**
The URL to the JNDI provider in WebLogic Network Gatekeeper.
Example: t3://<host>:<port>

**Attribute: Username**
A WebLogic Network Gatekeeper administrative user.

**Attribute: Password**
The password associated with the WebLogic Network Gatekeeper administrative user.

**Attribute: Controller URI**
The Controller SIP URI that is used to establish the third party call. If this value is set, a call appears to the callee to come from this URI. By default, the value is “None”, where no controller URI will be used to establish the call. In this case, the call appears to the callee to come from the caller.

**Attribute: Suggested subscription lifetime**
Specifies a suggested lifetime for a presence subscription, given in seconds.
This value might not be accepted by the Presence Server. In this case the Presence Server will set the expiry value it has chosen, and the value to use, in the first NOTIFY sent to the WebLogic SIP Server presence plug-in. The lifetime for the presence subscription will be according to the value received from the Presence Server.

**Attribute: Presence Server address**
The address to which the subscribe messages are sent. It can be the IP of the presence server or another IMS node that proxies the request.
The value is a SIP URI, for example sip:<host>:<port>
Managing and Configuring the Trace Service

The following section describes how to configure and manage the Trace Service.

Introduction to the Trace Service

The trace service is based on Log4J. Each Network Gatekeeper service maintain its own log file, named <service name>.log and each service instance writes its log file to the local file system of the server it executes on. The trace service writes log files in the directory

$DOMAIN_HOME/servers/<server name>/trace

Basic tracing

For basic tracing, the root logger rootAT maintains the trace file defaultAT.log and the root logger rootNT maintains the trace file defaultNT.log.

Below is a list of attributes and operations for configuration and maintenance for basic tracing:

- Attribute: TracingEnabled (r)
- Attribute: BytebufferSize
- Operation: attachAppender
- Operation: attachAppenderInternal
- Operation: createTraceDir
- Operation: disableTracing
Managing and Configuring the Trace Service

- **Operation: enableTracing**
- **Operation: flushBuffers**
- **Operation: rollOver**

**Context trace**

In addition to basic tracing, context tracing generates log messages filtered on the context of a request, for example a certain service provider or application. New context trace files can be added, and context filters and context categories can be applied to these files. Context categories can be added to the context trace file in order to log messages from one or more Network Gatekeeper services. For example, to log messages from the budget service and the SMPP plug-in to a context trace file, the context categories for these are added to the context trace file. Context trace filters narrows the generated log messages to requests that matches a context filter. A context filter has pre-defined filter types that defines what to filter on. For a given filter type, the value to match is defined. This is used for tracing on individual service providers, application and so on.

The workflow for defining a context trace file is:

1. In Network Gatekeeper Administration Console choose **Services→Trace Service**
2. **Operation: createContextTraceFile**, or **Operation: createRootContextTraceFile**.
3. For each context category to add for the context trace file, use operation **Operation: addContextCategory**.
4. For each context filter to add, use **Operation: addContextFilter**

Below is a list of attributes and operations for configuration and maintenance for context tracing:

- **Operation: addContextCategory**
- **Operation: addContextFilter**
- **Operation: createContextTraceFile**
- **Operation: createRootContextTraceFile**
- **Operation: removeContextTraceFile**
- **Operation: resetContextFilters**
Reference: attributes and operations for Trace Service

**Attribute: TracingEnabled (r)**
Scope: Cluster
Read-only.
True if tracing is enabled, otherwise False.

**Attribute: ByteBufferSize**
Scope: Cluster
Unit: Byte
Sets the buffer size for the trace.

**Operation: addContextCategory**
Scope: Cluster
Adds a Log4J context category.
Signature:
addContextCategory(identifier: String, category: String)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>identifier</td>
<td>ID of a context trace file.</td>
</tr>
<tr>
<td>category</td>
<td>The Log4J category. The category is the package name, including sub-packages, to trace on. For example, to trace on the SMPP plug-in, enter com.bea.wlcp.wlng.plugin.sms.smpp</td>
</tr>
</tbody>
</table>
Operation: addContextFilter

Scope: Cluster

Adds one of the pre-defined filter types to the appender with the identified name.

The filter is a name-value pair, where the type identifies the filter type to use, and the value is the value of the filter.

All trace information that matches the filter is written to the context trace file.

When the trace service receives information about a request, it checks if the request matches the context filter. If it matches, the trace information is written to file. All filters must match for the trace to be written.

Examples:

To add a filter that matches all requests from service provider ID SP1, the type shall be defined as SERVICE_PROVIDER and the value shall be set to SP1.

To add a filter that matches all requests from application ID APP1, the type shall be defined as APPLICATION and the value shall be set to APP1.

Signature:

addContextFilter(identifier: String, type: String, value: String)

Table 14-2  addContextFilter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>identifier</td>
<td>ID of a context trace file.</td>
</tr>
</tbody>
</table>
Operation: attachAppender

Scope: Cluster

Each Network Gatekeeper service has a default FileAppender that outputs to a file with the same name as the Network Gatekeeper service. This method allows adding named appender to named loggers.

Signature:

attachAppender(loggerName: String, sleeSvcName: String)

Table 14-3 attachAppender

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loggerName</td>
<td>Name of logger.</td>
</tr>
<tr>
<td>sleeSvcName</td>
<td>Name of Network Gatekeeper service to attach the appender to.</td>
</tr>
</tbody>
</table>

Operation: attachAppenderInternal

Scope: Cluster
Managing and Configuring the Trace Service

For internal use only.

**Operation: createContextTraceFile**

Scope: Cluster

Create a new context trace file.

Signature:

```java
createContextTraceFile(sleeSvcName: String, category: String, threshold: String)
```

**Table 14-4 createContextTraceFile**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>identifier</td>
<td>ID of the context trace file to create. Also the name of trace file. The filename has the suffix .log. Must be unique per managed server.</td>
</tr>
<tr>
<td>category</td>
<td>The Log4J category.</td>
</tr>
</tbody>
</table>
| threshold | The threshold for the context trace file:  
  - ERROR  
  - WARN  
  - INFO  
  - DEBUG  
  - LOG |

**Operation: createRootContextTraceFile**

Scope: Cluster

Create a new root context trace file. Adds a context trace file under root trace directory and associates it with the root logger.

Signature:

```java
createRootContextTraceFile(identifier: String)
```
Operation: createTraceDir
Internal use only.

Operation: disableTracing
Scope: Cluster
Disable all tracing.
Signature:
disableTracing()

Table 14-5 createRootContextTraceFile

createContextTraceFile

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>identifier</td>
<td>ID of the context trace file. Also the name of trace file. The filename has the suffix .log. Must be unique per managed server.</td>
</tr>
</tbody>
</table>

Operation: enableTracing
Scope: Cluster
Enable all tracing.
Signature:
enableTracing()
Managing and Configuring the Trace Service

### Operation: flushBuffers

**Scope:** Cluster

Flush trace buffers to file. This method only has effect on FileAppenders or subclasses.

**Signature:**

```java
flushBuffers()
```

### Table 14-7 enableTracing

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 14-8 flushBuffers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Operation: initialize

**Scope:** Cluster

Internal use only.

### Operation: removeContextTraceFile

**Scope:** Cluster

Removes a context trace file.

**Signature:**

```java
removeContextTraceFile(identifier: String)
```
**Operation: resetContextFilters**

Scope: Cluster

Removes all filters associated with a context trace file.

Signature: 

rollOver()

**Table 14-9 removeContextTraceFile**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>identifier</td>
<td>ID of context trace file to remove.</td>
</tr>
</tbody>
</table>

**Operation: rollOver**

Scope: Cluster

Aggregate method for RollingFileAppenders, calls rollOver() on all registered appenders

Signature: 

rollOver()
Log4J Hierarchies, loggers, and appenders

There is a set of Log4J Dynamic MBeans shipped with appenders that comes by default with Network Gatekeeper. Two Log4J hierarchies are defined: AT (for the access tier) and NT (for the network tier): these are displayed in the following entries in the Network Gatekeeper management console under Log4J:

- log4j:hierarchy=AT
- log4j:hierarchy=NT

To the AT hierarchy, the logger rootAT is connected, also displayed in the list:

- log4j:Location=AdminServer,logger=rootAT

To the NT hierarchy, the logger rootNT is connected, also displayed in the list:

- log4j:Location=AdminServer,logger=rootNT

Additional loggers can be added, and additional appenders can be added to the loggers. It is possible to change the priority of the logger and which appender to use. Parameters for the loggers and appenders can be configured.

A set of default appenders are defined:

- log4j:appender=defaultAT,Location=AdminServer
- log4j:appender=defaultAT,Location=AdminServer,layout=org.apache.log4j.TTCCLayout
- log4j:appender=defaultNT,Location=AdminServer
- log4j:appender=defaultNT,Location=AdminServer,layout=org.apache.log4j.TTCCLayout

Note: The Log4J attributes are dynamic and not persisted.
Configuring Trace for Access Tier servers

Trace configuration for network tier servers is performed via the Network Gatekeeper Administration console, in `<Server Name> -> Log4J`. For Access tier servers, there is no management attributes or operations exposed in the Network Gatekeeper Administration console. This configuration must be done directly on the MBeans using a JMX-based management console such as JConsole or using WLST.

For example, if using WLST, connect to the MBean server as described in Chapter 30, “Writing OAM Scripts and using JMX.” and change to the custom tree where Network Gatekeeper MBeans are located. Change to the Log4J directory: `cd('log4J')`

The settings for the access tier servers can be changed in the following directories:

- `log4j:appender=defaultAT`
- `log4j:appender=defaultAT,layout=org.apache.log4j.TTCCLayout`
- `log4j:hiearchy=AT`
- `log4j:logger=rootAT`

For example, to change the trace level for the access tier servers for the appender `log4j:appender=defaultAT` to WARN:

1. `cd('log4j:appender=defaultAT')`
2. `set('threshold','WARN')`
Managing and Configuring the Trace Service
Managing and Configuring Third Party Call Traffic Paths

The following section describes configuration and maintenance attributes and operations for the Parlay X 2.1 Third Party call traffic paths. They also provide a workflow for the configuration. There are two main types of traffic paths for Third Party Call available:

- The Enhanced Parlay X 2.1 Third Party Call traffic path with SIP integration
- The Backwards Compatible Third Party Call Traffic Path with OSA MultiParty Call Control
  - Stateless adapter for the backwards compatible Third Party Call traffic path
  - Service capability for the backwards compatible Third Party Call traffic path
  - OSA MultiParty Call Control plug-in for the backwards compatible Parlay X 2.1 Third Party Call traffic path

Third Party Call Traffic Paths

The Third Party Call traffic paths are of both the enhanced and backwards compatible types, depending on the network protocol to which they connect.

The Enhanced Parlay X 2.1 Third Party Call traffic path with SIP integration

This section contains a description of the configurations attributes and operations available for the SIP integration for the Third Party Call traffic path.
Network Gatekeeper uses two parts to integrate with SIP networks, a part that executes in Network Gatekeeper, and a part that executes as a SIP Application in BEA WebLogic SIP Server. The two parts execute in different containers and must be configured in both.

Configuration for the SIP Application part is described in Configuring WebLogic SIP Server Integration.

For a configuration workflow, see Configuration workflow.

### Configuration workflow

1. In Network Gatekeeper Management Console, select the managed object `Plugin_third_party_call_sip`.
2. Configure connection information for the connection to WebLogic SIP Server:
   - Operation: `setSIPNodeURL`
   - Operation: `setSIPNodeUserName`
   - Operation: `setSIPNodePassword`

   **Note:** The user must be registered in WebLogic SIP Server.

3. Configure behavior of the network protocol plug-in:
   - Operation: `setChargingAllowed`
   - Operation: `setStatusRetentionTime`

4. Setup the routing rules to the plug-in(s), see Administration of plug-in routes. The plug-in ID is `Plugin_third_party_call_sip`.

5. If desired, create and load a node SLA, see Updating and loading a global node SLA.

Move on to provisioning of service provider accounts and application accounts.

### Reference: attributes and operations for Plugin_third_party_call_sip

Below is a list of attributes and operations for configuration and maintenance.

- **Attribute:** `StatusRetentionTime (r)`
- **Attribute:** `ChargingAllowed (r)`
- **Attribute:** `SIPNodePassword (r)`
- **Attribute:** `SIPNodeUsername (r)`
- **Attribute: SIPNodeUrl**

**Attribute: StatusRetentionTime (r)**
Scope: Cluster
Unit: seconds
Format: int
Displays the call status retention time.

**Attribute: ChargingAllowed (r)**
Scope: Cluster
Unit: n/a
Format: boolean
Displays if charging is allowed or not.

**Attribute: SIPNodePassword (r)**
Scope: Cluster
Unit: n/a
Format: String
Displays the password for the WebLogic SIP Server JNDI access username of the application part of the SIP integration for Call Notification. This is a regular WebLogic SIP Server user.

**Attribute: SIPNodeUsername (r)**
Scope: Cluster
Unit: n/a
Format: String
Displays the WebLogic SIP Server JNDI access username for the SIP application part of the SIP integration for Call Notification. This is a regular WebLogic SIP Server user.

**Attribute: SIPNodeUrl**
Scope: Cluster
Unit: n/a
Format: String in URI format
Displays the WebLogic SIP Server naming service containing the host name (or IP address) and the port number identifying `wlss.jndi.uri`. This was configured when WebLogic SIP Server was installed. The WebLogic SIP Server JNDI URL.
Example: `t3://127.0.0.1:7002`

**Note:** This is not the same as the Network Gatekeeper naming service

**Operation: setChargingAllowed**
Scope: Cluster
Specifies if charging is allowed or not.
Signature:
```
setChargingAllowed(p1: boolean)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Use true if charging shall be allowed, otherwise false.</td>
</tr>
</tbody>
</table>

**Operation: setSIPNodePassword**
Scope: Cluster
Sets the password for the SIP node user.
Signature:
```
setSIPNodePassword(password: String)
```
Operation: setSIPNodeURL
Scope: Cluster

Specifies the WebLogic SIP Server naming service containing the host name (or IP address) and the port number identifying \texttt{wlss.jndi.uri}. This was configured when WebLogic SIP Server was installed. The WebLogic SIP Server JNDI URL.

Example: \texttt{t3://127.0.0.1:7002}

Note: This is not the same as the Network Gatekeeper naming service

Signature:

\texttt{setSIPNodeURL(p1: String)}

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>WebLogic SIP Server JNDI URL in URI format.</td>
</tr>
</tbody>
</table>

Operation: setSIPNodeUserName
Scope: Cluster

Specifies the WebLogic SIP Server JNDI access username for the SIP application part of the SIP integration for ThirdParty Call. This is a regular WebLogic SIP Server user.

Signature:

\texttt{setSIPNodeUserName(p1: String)}
Managing and Configuring Third Party Call Traffic Paths

Table 15-4  setSIPNodePassword

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setSIPNodeUserName</td>
<td>Username.</td>
</tr>
</tbody>
</table>

Table 15-5  setStatusRetentionTime

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>The call status retention time.</td>
</tr>
</tbody>
</table>

**Operation: setStatusRetentionTime**

Scope: Cluster

Specifies the call status retention time.

Signature:

setStatusRetentionTime(p1: int)

### The Backwards Compatible Third Party Call Traffic Path with OSA MultiParty Call Control

The backwards compatible Third Party Call traffic path uses an entire set of modules that must be configured, including the network plug-in that performs the protocol-specific communication with the underlying network node. The following common modules need to be configured:

- Stateless adapter for the backwards compatible Third Party Call traffic path
- Service capability for the backwards compatible Third Party Call traffic path

The following network plug-ins are provided:

- OSA MultiParty Call Control plug-in for the backwards compatible Parlay X 2.1 Third Party Call traffic path
Add the traffic path to Network tier

The backwards compatible traffic path for Third Party call is not installed by default.

Follow the steps below to add the traffic path to the network tier:

1. Copy the network tier ear file to a temporary directory, $tmp. The network tier ear file is found in $DOMAIN_HOME/wlng_nt.ear.

2. Explode the ear file using unzip or other tool.

3. Copy the following files to $tmp:
   
   $DOMAIN_HOME/bc/c_osa_nichs.jar
   $DOMAIN_HOME/bc/resource_osa_mpcc.jar
   $DOMAIN_HOME/bc/x_espa_call_control.jar
   $DOMAIN_HOME/bc/x_sespa_callcontrol.jar
   $DOMAIN_HOME/bc/y_tpc_retained.jar

4. Edit the file $tmp/META-INF/weblogic-extension.xml.

5. For each of the files listed above, add the following:

   <custom-module>
   
   <uri><name of jar-file to add></uri>
   <provider-name>SleeModule</provider-name>
   
   </custom-module>

   Where <name of jar-file to add> is the name of the jar file listed earlier. It is important that these are added last in weblogic-extension.xml and that the order is as follows.

   <custom-module>
   
   <uri>y_tpc_retained.jar</uri>
   <provider-name>SleeModule</provider-name>
   
   </custom-module>

   <custom-module>
   
   <uri>c_osa_nichs.jar</uri>
   <provider-name>SleeModule</provider-name>
   
   </custom-module>

   <custom-module>
   
   </custom-module>
Managing and Configuring Third Party Call Traffic Paths

6. Remove the SIP plug-in for Third Party Call, since the two plug-ins cannot be installed in parallel. In weblogic-extension.xml, find the following entry and remove it:

   <custom-module>
     <uri>j_tpc_sip.jar</uri>
     <provider-name>SleeModule</provider-name>
   </custom-module>

7. Package the directory $tmp, including subdirectories, to wlng_nt.ear. This can, for example be done with zip:

   ```
   zip -r wlng_nt.ear $tmp
   ```

8. Copy the newly created wlng_nt.ear to $DOMAIN_HOME.

9. Redeploy wlng_nt.ear.

Configuration workflow

Below is an outline for configuring the backwards compatible Third Party Call traffic path using Network Gatekeeper Management Console:

1. Setup routing rules to the backwards compatible plug-in(), see Administration of plug-in routes. The plug-in ID to use is Plugin_third_party_call_bc.

2. Decide the behavior of the traffic path by configuring the Third Party Call stateless adapter. The following configuration parameters are available:
   - Operation: setStorageSetting in the managed object SESPA_callcontrol.
– Operation: disableAddressPresentation or Operation: enableAddressPresentation in the managed object ESPA_callcontrol.
– Operation: generateCallInformation in the managed object ESPA_callcontrol.
– Operation: setPlanCallingAddress in the managed object ESPA_callcontrol.
– Attribute: OverloadPercentage
– Attribute: RequestTimeout
– Attribute: SevereOverloadPercentage

3. Define which network protocol plug-ins to use and follow the configuration flow for each plug-in:
   – OSA MultiParty Call Control plug-in for the backwards compatible Parlay X 2.1 Third Party Call traffic path

4. Setup the routing rules to the plug-in(s), see Administration of plug-in routes. The plug-in ID to use is generated.

5. If desired, create and load a node SLA, see Updating and loading a global node SLA. When you are done, you can move on to the provisioning of service provider accounts and application accounts.

Stateless adapter for the backwards compatible Third Party Call traffic path

This section contains a description of the configuration attributes and operations available for the Stateless adapter part of the backwards compatible Third Party Call traffic path.

Reference: attributes and operations for SESPA_callcontrol

All attributes and operations are available in the managed object SESPA_callcontrol:

Attribute: StorageSettings (r)
Operation: addParlayXNetworkCallListener
Operation: getParlayXNetworkCallListener
Operation: removeParlayXNetworkCallListener
Operation: setStorageSetting
Attribute: ParlayXNetworkCallListeners (r)
**Attribute: StorageSettings (r)**
Displays the database storage settings for registered notifications.
Unused in the context of the Backwards Compatible Third Party Call Traffic Path with OSA MultiParty Call Control.

**Operation: addParlayXNetworkCallListener**
Adds a Parlay X network call listener. An ID for the listener is returned.
Unused in the context of the Backwards Compatible Third Party Call Traffic Path with OSA MultiParty Call Control.

Signature:
```
addParlayXNetworkCallListener(params.endpoint: String,
params.serviceProviderID: String, params.applicationID: String,
params.applicationInstanceGroupID: String, params.aPartyAddressExpression: String,
params.bPartyAddressExpression: String, params.callEventCriteria: int,
params.serviceCode: String, params.requesterID: String)
```

**Table 15-6  addParlayXNetworkCallListener**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint</td>
<td>The URL to the application endpoint.</td>
</tr>
<tr>
<td>serviceProviderID</td>
<td>Service provider account ID.</td>
</tr>
<tr>
<td>applicationID</td>
<td>Application account ID.</td>
</tr>
<tr>
<td>applicationInstanceGroupID</td>
<td>Application instance group ID.</td>
</tr>
<tr>
<td>aPartyAddressExpression</td>
<td>A party expression to match. For example “tel:46*”</td>
</tr>
<tr>
<td>bPartyAddressExpression</td>
<td>B party expression to match. For example “tel:46*”</td>
</tr>
</tbody>
</table>
Operation: `getParlayXNetworkCallListener`

Displays detailed information about a registered Parlay X network call listener.

Unused in the context of the Backwards Compatible Third Party Call Traffic Path with OSA MultiParty Call Control.

Signature:

`getParlayXNetworkCallListener(id:long)`

---

**Table 15-6  addParlayXNetworkCallListener**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callEventCriteria</td>
<td>Call event ID. The call event ID defines which call state should trigger the event and the notification to be sent. Use:</td>
</tr>
<tr>
<td></td>
<td>• 0 for call state ADDRESS_ANALYSED</td>
</tr>
<tr>
<td></td>
<td>• 1 for call state ORIGINATING_CALL_AUTHORISED</td>
</tr>
<tr>
<td></td>
<td>• 2 for call state TERMINATING_CALL_AUTHORISED</td>
</tr>
<tr>
<td></td>
<td>• 3 for call state ORIGINATING_CALL_ATTEMPT</td>
</tr>
<tr>
<td></td>
<td>• 4 for call state TERMINATING_CALL_ATTEMPT</td>
</tr>
<tr>
<td></td>
<td>• 5 for call state ALERTING (not supported by Parlay X)</td>
</tr>
<tr>
<td></td>
<td>• 6 for call state BUSY</td>
</tr>
<tr>
<td></td>
<td>• 7 for call state NO_ANSWER</td>
</tr>
<tr>
<td></td>
<td>• 8 for call state CALL_FAILURE</td>
</tr>
<tr>
<td></td>
<td>• 9 for call state CALL_ANSWERED (not supported by Parlay X)</td>
</tr>
<tr>
<td></td>
<td>• 10 for call state DISCONNECT (not supported by Parlay X)</td>
</tr>
<tr>
<td>serviceCode</td>
<td>The service issuing the request. If unknown/unspecified use empty string.</td>
</tr>
<tr>
<td>requesterID</td>
<td>ID for the requester of this service. If unknown/unspecified use empty string.</td>
</tr>
</tbody>
</table>
Managing and Configuring Third Party Call Traffic Paths

### Operation: removeParlayXNetworkCallListener

Removes a registered Parlay X network call listener.

Unused in the context of the Backwards Compatible Third Party Call Traffic Path with OSA MultiParty Call Control.

**Signature:**

```
removeParlayXNetworkCallListener(id: long)
```

### Table 15-7 getParlayXNetworkCallListener

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the listener. The ID was returned by Operation: addParlayXNetworkCallListener. See Attribute: ParlayXNetworkCallListeners (r) for a list.</td>
</tr>
</tbody>
</table>

### Operation: setStorageSetting

Sets database storage settings.

Unused in the context of the Backwards Compatible Third Party Call Traffic Path with OSA MultiParty Call Control.

**Signature:**

```
setStorageSetting(storageDurationDays: int, removalTime: String, removalSize: int)
```
Attribute: ParlayXNetworkCallListeners (r)

Read-only.

List of all enabled Parlay X listeners.

Unused in the context of the Backwards Compatible Third Party Call Traffic Path with OSA MultiParty Call Control.

Service capability for the backwards compatible Third Party Call traffic path

This section contains a description of the configurations attributes and operations available for the service capability part of the backwards compatible Third Party Call traffic path.

All attributes and operations are available in the managed object ESPA_callcontrol.

Reference: attributes and operations for ESPA_callcontrol

Attribute: OverloadPercentage

Attribute: RequestTimeout

Attribute: SevereOverloadPercentage

Operation: disableAddressPresentation
Operation: enableAddressPresentation
Operation: generateCallInformation
Operation: isCallInformationGenerated
Operation: setPlanCallingAddress
Operation: deleteNotification
Operation: listCallbacksForNotification
Operation: listNotifications

**Attribute: OverloadPercentage**
Scope: Server
Unit: n/a
Format: int [0...100]
Specifies the load percentage defining at what point the software module should raise an overloaded alarm. Must be less than **Attribute: SevereOverloadPercentage**.

**Attribute: RequestTimeout**
Scope: Cluster
Unit: n/a
Format: int
Sets the time-out in seconds for the requests made on the interface of the module.

**Attribute: SevereOverloadPercentage**
Scope: Server
Unit: n/a
Format: int [0...100]
Specifies the load percentage defining at what point the software module should raise a severely overloaded alarm.
Must be larger than **Attribute: OverloadPercentage**.
**Operation: disableAddressPresentation**

Scope: server

Disables address presentation, that is, addresses are not allowed to be presented to network users.

Signature:

disableAddressPresentation()

---

**Table 15-10  disableAddressPresentation**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disableAddressPresentation</td>
<td>-</td>
</tr>
</tbody>
</table>

---

**Operation: enableAddressPresentation**

Scope: server

Enables address presentation, that is, addresses are allowed to be presented to network users.

Signature:

enableAddressPresentation()

---

**Table 15-11  enableAddressPresentation**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enableAddressPresentation</td>
<td>-</td>
</tr>
</tbody>
</table>

---

**Operation: generateCallInformation**

Scope: server

Specifies if call information should be generated by Network Gatekeeper or if this information should be provided by the network.
Note: If configured to use information from the network and the network returns an error when
the requesting it, the information will still be generated by Network Gatekeeper.

Signature:
generateCallInformation(generateCallInfo: boolean)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>generateCallInfo</td>
<td>Use:</td>
</tr>
<tr>
<td></td>
<td>• true to let Network Gatekeeper generate the call information.</td>
</tr>
<tr>
<td></td>
<td>• false to let the network generate the call information.</td>
</tr>
</tbody>
</table>

**Operation: isCallInformationGenerated**

Scope: server

Specifies whether call information is generated by Network Gatekeeper or provided by the
network. The operation returns:

• true if call information is generated by Network Gatekeeper

• false if call information is provided by the network

Signature:
isCallInformationGenerated()
Operation: setPlanCallingAddress
Scope: Cluster
Sets the originating address. This address will be used as calling address. This method should be called once for every address plan.

Signature:
setPlanCallingAddress(addressURI: String)

Table 15-14 setPlanCallingAddress

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addressURI</td>
<td>Calling address in URI format.</td>
</tr>
</tbody>
</table>

Operation: deleteNotification
Scope: Cluster
Deletes a notification. A notification is registered off-line or by an application.

Unused in the context of the Backwards Compatible Third Party Call Traffic Path with OSA MultiParty Call Control.

Signature:
deleteNotification(id: int)

Table 15-15 deleteNotification

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID for the notification to remove.</td>
</tr>
</tbody>
</table>

Operation: listCallbacksForNotification
Scope: Server
Lists the callback interfaces for a notification.

Unused in the context of the Backwards Compatible Third Party Call Traffic Path with OSA MultiParty Call Control.

Signature:

\[
\text{listCallbacksForNotification}(\text{notificationId}: \text{int})
\]

### Table 15-16 listCallbacksForNotification

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notificationId</td>
<td>The ID of the notification.</td>
</tr>
</tbody>
</table>

**Operation: listNotifications**

Scope: Cluster

Lists information about existing notifications.

Signature:

\[
\text{listNotifications}(\text{clientId}: \text{String}, \text{startIndex}: \text{int}, \text{maxCount}: \text{int})
\]

### Table 15-17 listNotifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clientId</td>
<td>The client id or an empty string for no filtering.</td>
</tr>
<tr>
<td>startIndex</td>
<td>Start index of the list.</td>
</tr>
<tr>
<td>maxCount</td>
<td>The number of notifications in the list. Use 0 for all notifications.</td>
</tr>
</tbody>
</table>
OSA MultiParty Call Control plug-in for the backwards compatible Parlay X 2.1 Third Party Call traffic path

This section contains a description of the configuration attributes and operations available for the OSA MultiParty Call Control plug-in part of the backwards compatible Third Party Call traffic path. All attributes and operations are available in the managed object Plugin_OSA_MPCC.

Most of the configuration for the OSA Multiparty Call Control plug-in for the backwards compatible Third party Call traffic path is done in the OSA Access module, but with configuration details for the OSA Multiparty Call Control SCS. See Managing and Configuring OSA/Parlay Gateway Connections.

Configuration workflow

Below outlines how to configure the plug-in using the Network Gatekeeper Management Console:

1. Gather information about the OSA Gateway and configure the plug-in accordingly. The following information needs to be obtained from the OSA Gateway administrator and configured in the OSA Access service:
   - OSA/Parlay SCS type to be used in the look up (service discovery) phase when requesting the service (OSA/Parlay SCS) from the OSA/Parlay Gateway. Typically this is P_MULTI_PARTY_CALL_CONTROL.
   - OSA/Parlay service properties to be used in the look up (service discovery) phase when requesting a service (OSA/Parlay SCS) from the OSA/Parlay Gateway. This depends on the OSA Gateway implementation.
   - Authentication type used by the OSA/Parlay Framework.
   - Encryption method used for the connection with the OSA Gateway.
   - Signing algorithm used when signing the service level agreement with the OSA/Parlay Framework.

2. Setup the OSA Client and the OSA Client Mappings according to Creating an OSA client and Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS.

3. Setup the routing rules to the plug-ins, see Administration of plug-in routes.

4. If desired, create and load a node SLA, see Updating and loading a global node SLA.
**Reference: attributes and operations for Plugin_OSA_MPCC**

Below is a list of attributes and operations for configuration and maintenance:

**Attribute: NumberOfActiveSessions (r)**

Read-only.

Displays the number of active sessions towards the MultiParty Call Control SCS.
Managing and Configuring Call Notification Traffic Paths

The following section describes configuration and maintenance attributes and operations for the Parlay X 2.1 Call Notifications traffic paths. They also provide a workflow for the configuration:

- **OSA MultiParty Call Control for Call Notification traffic path**
  - Configuration workflow
  - Reference: attributes and operations for Plugin_call_notification_parlay_mpcc

- **SIP integration for the Parlay X 2.1 Call Notification traffic path**
  - Configuration workflow
  - Reference: Attributes and operations for Plugin_call_notification_sip

### Call Notification Traffic Paths

**OSA MultiParty Call Control for Call Notification traffic path**

For a configuration workflow, see Configuration workflow.

Most of the configuration for the OSA MultiParty Call Control plug-in for Call Notification is done in the OSA Access module, but with configuration details for the OSA MultiParty Call Control. See Managing and Configuring OSA/Parlay Gateway Connections

This section contains a description of the configuration attributes and operations available for OSA MultiParty Call Control for the Call Notification traffic path.
Configuration workflow

1. Gather information about the OSA Gateway and configure the MultiParty Call Control part of the plug-in accordingly. The following information needs to be obtained from the OSA Gateway administrator and configured in the OSA Access service:

   - OSA/Parlay SCS type to be used in the look up (service discovery) phase when requesting the MultiParty Call Control service (OSA/Parlay SCS) from the OSA/Parlay Gateway. Typically this is `P_MULTI_PARTY_CALL_CONTROL`.
   - OSA/Parlay service properties to be used in the look up (service discovery) phase when requesting a service (OSA/Parlay SCS) from the OSA/Parlay Gateway. This depends on the OSA Gateway implementation.
   - Authentication type used by the OSA/Parlay Framework.
   - Encryption method used for the connection with the OSA Gateway.
   - Signing algorithm used when signing the service level agreement with the OSA/Parlay Framework.

2. Setup the OSA Client and the OSA Client Mappings according to Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS in section Managing and Configuring OSA/Parlay Gateway Connections

3. Setup the routing rules to the plug-in, see Administration of plug-in routes. Use the plug-in ID `Plugin_call_notification_parlay_mpcc`.

4. If desired, create and load a node SLA, see Updating and loading a global node SLA.

Management and provisioning operations

The following operations are related to provisioning and management:

- **Operation:** deleteNotification
- **Operation:** getNotification
- **Operation:** listNotifications

Reference: attributes and operations for `Plugin_call_notification_parlay_mpcc`

Below is a list of attributes and operations for configuration and maintenance:

- **Operation:** deleteNotification
• Operation: getNotification
• Operation: listNotifications

**Operation: deleteNotification**

Scope: Cluster

Deletes a notification.

Signature:

```java
deleteNotification(correlator: String)
```

<table>
<thead>
<tr>
<th>Table 16-1 deleteNotification</th>
</tr>
</thead>
<tbody>
<tr>
<td>deleteNotification</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>ID for the subscription. Given by an application when the subscription was started.</td>
</tr>
</tbody>
</table>

**Operation: getNotification**

Scope: Cluster

Displays information about a notification.

Signature:

```java
getNotification(correlator: String)
```

<table>
<thead>
<tr>
<th>Table 16-2 getNotification</th>
</tr>
</thead>
<tbody>
<tr>
<td>getNotification</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>ID for the subscription. Given by an application when the subscription was started.</td>
</tr>
</tbody>
</table>
Managing and Configuring Call Notification Traffic Paths

**Operation: listNotifications**

Scope: Cluster

Displays all registered notifications.

Signature:

`listNotification()`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**SIP integration for the Parlay X 2.1 Call Notification traffic path**

This section contains a description of the configuration attributes and operations available for the SIP integration for the Call Notification traffic path.

Network Gatekeeper uses two parts to integrate with SIP networks:

One part executes one Network Gatekeeper, and one part executes as a SIP Application in BEA WebLogic SIP Server. The two parts execute in different containers and require configuration in both.

Configuration for the SIP Application part is described in Configuring WebLogic SIP Server Integration.

For a configuration workflow, see Configuration workflow.

**Configuration workflow**

1. There is no specific configuration to setup in Plugin_call_notification_sip.

2. Setup the routing rules to the plug-in(s), see Administration of plug-in routes. The plug-in ID is Plugin_call_notification_sip.
3. If desired, create and load a node SLA, see Updating and loading a global node SLA. Move on to the provisioning of service provider accounts and application accounts.

**Reference: Attributes and operations for Plugin_call_notification_sip**

Below is a list of attributes and operations for provisioning and maintenance:

- Operation: `getCallDirectionSubscription`
- Operation: `getNotificationSubscription`
- Operation: `listCallDirectionSubscriptions`
- Operation: `listNotificationSubscriptions`
- Operation: `removeAllCallDirectionSubscriptions`
- Operation: `removeAllNotificationSubscriptions`
- Operation: `removeCallDirectionSubscription`
- Operation: `removeNotificationSubscription`

**Operation: getCallDirectionSubscription**

Scope: Cluster

Displays call direction subscription information.

**Signature:**

`getCallDirectionSubscription(correlator: String)`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>ID for the subscription. Given by an application when the subscription was started.</td>
</tr>
</tbody>
</table>

**Operation: getNotificationSubscription**

Scope: Cluster
Displays call notification subscription information.

Signature:

getNotificationSubscription(correlator: String)

---

### Table 16-5 getCallDirectionSubscription

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>ID for the subscription. Given by an application when the subscription was</td>
</tr>
<tr>
<td></td>
<td>started.</td>
</tr>
</tbody>
</table>

---

**Operation: listCallDirectionSubscriptions**

Scope: Cluster

Displays a list of correlators for call direction subscriptions.

Signature:

listCallDirectionSubscriptions(offset: int, length: int)

---

### Table 16-6 listCallDirectionSubscriptions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>offset</td>
<td>Start of offset.</td>
</tr>
<tr>
<td>length</td>
<td>Number of entries returned.</td>
</tr>
</tbody>
</table>

---

**Operation: listNotificationSubscriptions**

Scope: Cluster

Displays a list of correlators for call notification subscriptions.

Signature:

listNotificationSubscriptions(offset: int, length: int)
Table 16-7  listNotificationSubscriptions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>offset</td>
<td>Start of offset.</td>
</tr>
<tr>
<td>length</td>
<td>Number of entries returned.</td>
</tr>
</tbody>
</table>

**Operation: removeAllCallDirectionSubscriptions**

Scope: Cluster

Removes all call direction subscriptions.

Signature:

`removeAllCallDirectionSubscriptions()`

Table 16-8  removeAllCallDirectionSubscriptions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: removeAllNotificationSubscriptions**

Scope: Cluster

Removes all call notification subscriptions.

Signature:

`removeAllNotificationSubscriptions()`
Managing and Configuring Call Notification Traffic Paths

Table 16-9  removeAllNotificationSubscriptions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Operation: removeCallDirectionSubscription**

Scope: Cluster

Removes a call direction subscription.

Signature:

```
removeCallDirectionSubscription(correlator: String)
```

Table 16-10  removeCallDirectionSubscription

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>ID for the subscription. Given by an application when the subscription was started.</td>
</tr>
</tbody>
</table>

**Operation: removeNotificationSubscription**

Scope: Cluster

Removes a call notification subscription.

Signature:

```
removeNotificationSubscription(correlator: String)
```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>ID for the subscription. Given by an application when the subscription was started.</td>
</tr>
</tbody>
</table>
Managing and Configuring Short Messaging Traffic Paths

The following section describes configuration and maintenance attributes and operations for the Parlay X 2.1 Short Messaging traffic paths. They also provides a workflow for the configuration:

- SMPP plug-in for Parlay X 2.1 Short Messaging
  - Configuration workflow
  - Provisioning workflow
  - Management operations
  - Reference: attributes and operations for Plugin_sms_SMPP
- OSA Multimedia Messaging plug-in for Parlay X 2.1 Short Messaging
  - Configuration workflow
  - Provisioning workflow
  - Management operations
  - Reference: attributes and operations for Plugin_sms_parlay_mmm

Short Messaging Traffic Paths

**SMPP plug-in for Parlay X 2.1 Short Messaging**

Managed object: Plugin_sms_smpp

For a configuration workflow, see Configuration workflow.
Configuration workflow

Below is an outline on how to configure the plug-in using Network Gatekeeper Management Console:

1. In Plugin_sms_SMPP, configure the behavior of the plug-in by defining the following attributes:
   - Attribute: DefaultDataCoding
   - Attribute: EsmeAddressRange
   - Attribute: EsmeNpi
   - Attribute: EsmePassword
   - Attribute: EsmeSystemId
   - Attribute: EsmeSystemType
   - Attribute: EsmeTon
   - Attribute: RequestDeliveryReports
   - Attribute: ReceiverProxyLocalAddressAndPort
   - Attribute: SMSCDefaultAlphabet
   - Attribute: SequenceNumberRangeEndId
   - Attribute: SequenceNumberRangeStartId
   - Attribute: SmscAddress
   - Attribute: SmscPort
   - Attribute: UserTextMaxLength
   - Operation: addOriginatingAddressTypeMapping
   - Operation: translateOriginatingAddressNpi
   - Operation: translateOriginatingAddressTon
   - Attribute: ConnectDelayValue
   - Attribute: ConnectTotalTimeValue
   - Attribute: EnquireLinkRequestTimerValue
   - Attribute: EnquireLinkTimerValue
   - Attribute: RequestTimerValue
2. Setup the routing rules to the plug-in, see Administration of plug-in routes. Use the plug-in ID Plugin_sms_smpp.

3. If desired, create and load a node SLA, see Updating and loading a global node SLA. Continue with the provisioning of service provider accounts and application accounts.

**Provisioning workflow**
Below is an outline of provisioning the plug-in with service provider application specific data using Network Gatekeeper Management Console:

1. In Plugin_sms_SMPP, provision the following data:
   - Operation: addShortCodeTranslation
   - Operation: enableReceiveSms

**Management operations**
The following operations are related to management:

- Operation: countOfflineNotificationCache
- Operation: countOnlineNotificationCache
- Operation: countSmsCache
- Operation: getOfflineNotificationInfo
- Operation: getOnlineNotificationInfo
- Operation: removeOfflineNotificationInfo
- Operation: removeOnlineNotificationInfo
- Operation: resetSMPPConnection

**Parameters that can be set using contextAttribute**
The SMPP plug-in supports some parameters whose values can be defined in service provider and application-level SLAs. These parameters are passed in the request context and can be set from the SLA or a policy rule.
The example below illustrates an SLA that sets `com.bea.wlcp.wlng.plugin.sms.RequestDeliveryReportFlag` to true when the request originates from a service provider belonging to the service provider ID group `sp_grp_1`.

### Listing 17-1  Example: setting the value of a context attribute from a service provider group SLA

```xml
<xml version="1.0" encoding="UTF-8"?>
<sla serviceProviderGroupID="sp_grp_1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:namespace="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file://policy/sla_schema/sp_sla_file.xsd">
<serviceContract>
<startDate>2001-01-01</startDate>

Table 17-1  Supported context attributes

<table>
<thead>
<tr>
<th>attributeName</th>
<th>Parameter description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>com.bea.wlcp.wlng.plugin.sms.Priority</code></td>
<td>Message priority. A string with a value of either:</td>
</tr>
<tr>
<td></td>
<td>• HIGH</td>
</tr>
<tr>
<td></td>
<td>• LOW</td>
</tr>
<tr>
<td></td>
<td>• DEFAULT</td>
</tr>
<tr>
<td></td>
<td>• UNDEFINED</td>
</tr>
<tr>
<td><code>com.bea.wlcp.wlng.plugin.sms.RequestDeliveryReportFlag</code></td>
<td>Delivery report flag (a string with a Boolean value).</td>
</tr>
<tr>
<td></td>
<td>If set to false a delivery report for the sent message will not be requested and no</td>
</tr>
<tr>
<td></td>
<td>message metadata will be stored for future correlation with a delivery report.</td>
</tr>
<tr>
<td><code>com.bea.wlcp.wlng.plugin.sms.OriginatingAddressType</code></td>
<td>Originating address type (a string).</td>
</tr>
<tr>
<td></td>
<td>Should map to a value configured through OAM interface, see Operation:</td>
</tr>
<tr>
<td></td>
<td>addOriginatingAddressTypeMapping.</td>
</tr>
</tbody>
</table>
```
Reference: attributes and operations for Plugin_sms_SMPP

Below is a list of attributes and operations for configuration and maintenance:

- **Attribute**: ActiveStatus (r)
- **Attribute**: RequestDeliveryReports
- **Attribute**: DefaultDataCoding
- **Attribute**: SMSCDefaultAlphabet
- **Attribute**: MobileCountryCode
- **Attribute**: MobileNetworkCode
- **Attribute**: EnquireLinkRequestTimerValue
- **Attribute**: RequestTimerValue
- **Attribute**: SequenceNumberRangeEndId
- **Attribute**: SequenceNumberRangeStartId
- Attribute: ConnectDelayValue
- Attribute: ConnectTotalTimeValue
- Attribute: UserTextMaxLength
- Attribute: EnquireLinkTimerValue
- Attribute: SmppVersion (r)
- Attribute: EsmeNpi
- Attribute: EsmeTon
- Attribute: EsmeAddressRange
- Attribute: EsmeSystemType
- Attribute: EsmeSystemId
- Attribute: EsmePassword
- Attribute: SmscPort
- Attribute: SmscAddress
- Attribute: RequestDeliveryReports
- Attribute: TransmitterProxyLocalAddressAndPort
- Operation: addOriginatingAddressTypeMapping
- Operation: addShortCodeTranslation
- Operation: countOfflineNotificationCache
- Operation: countOnlineNotificationCache
- Operation: countSmsCache
- Operation: enableReceiveSms
- Operation: getOfflineNotificationInfo
- Operation: getOnlineNotificationInfo
- Operation: listOfflineNotificationInfo
- Operation: listOnlineNotificationInfo
• Operation: listOriginatingAddressTypeMappings
• Operation: listShortCodeTranslations
• Operation: removeOfflineNotificationInfo
• Operation: removeOnlineNotificationInfo
• Operation: removeOriginatingAddressTypeMapping
• Operation: removeShortCodeTranslation
• Operation: resetSMPPConnection
• Operation: startSmsNotification
• Operation: translateOriginatingAddressNpi
• Operation: translateOriginatingAddressTon

**Attribute: ActiveStatus (r)**
Read-only.
Scope: Server
Unit: n/a
Displays the state of the connection between the plug-in and the SMSC. True if the transmitter is successfully connected to the SMPP server, false if not.

**Attribute: ConnectDelayValue**
Scope: Server
Unit: seconds
Format: int
Specifies the delay time between connection attempts, when the connection to the SMSC is lost.

**Attribute: ConnectTotalTimeValue**
Scope: Server
Unit: minutes
Format: int
Specifies the amount of time to keep trying to connect to the SMSC, when the connection is lost.

**Attribute: DefaultDataCoding**
Scope: Server
Unit: n/a
Format: int
Specifies the default data coding to use when sending SMS messages. This value will be used if a data coding is not provided by the north interface.

See `data_coding` parameter in the SMPP specification for valid values. Use:

- 0 for SMSC Default Alphabet
- 1 for ASCII
- 8 for USC2

**Attribute: EnquireLinkRequestTimerValue**
Scope: Server
Unit: milliseconds
Format: int
Specifies the value of the timer used when sending Enquire Link requests.

**Attribute: EnquireLinkTimerValue**
Scope: Server
Unit: minutes
Format: int
Specifies the Enquire Link Timer value. The plug-in performs an Enquire Link operation to the SMSC to keep the connection alive. The time between enquiries is specified by this timer value.

**Note:** To turn off the sending of Enquire Link, set the timer value to 0.

**Attribute: EsmeAddressRange**
Scope: Server
Unit: n/a
Format: String formatted as a regular expression.
Specifies the ESME address range. Specifies the address range of the SMSs to be sent to the plug-in by the SMSC. The address range is specified as a UNIX regular expression.

**Attribute: EsmeNpi**
Scope: Server
Unit: n/a
Format: int
Specifies the ESME Numbering Plan Indicator (NPI).
Used for destination address and as a default for originating address. Also used for both destination address and originating address during bind operation. Use:

- 0 for Unknown
- 1 for ISDN (E163/E164)
- 3 for Data (X.121)
- 4 for Telex (F.69)
- 6 for Land Mobile (E.212)
- 8 for National
- 9 for Private
- 10 for ERMES
- 14 for Internet (IP)
- 18 for WAP Client ID

**Attribute: EsmePassword**
Scope: Server
Unit: n/a
Format: string
Specifies the password used by the plug-in when connecting to the SMSC as an ESME.
Attribute: EsmeSystemId
Scope: Server
Unit: n/a
Format: string
Specifies the system ID used by the plug-in when connecting to the SMSC as an ESME.

Attribute: EsmeSystemType
Scope: Server
Unit: n/a
Format: string
Specifies the system type used by the plug-in when connecting to the SMSC as an ESME.

Attribute: EsmeTon
Specifies the ESME Type Of Number (TON). Used for destination address and as a default for originating address. Also used for both destination address and originating address during bind operation. Use:

- 0 for UNKNOWN
- 1 for INTERNATIONAL
- 2 for NATIONAL
- 3 for NETWORK
- 4 for SUBSCRIBER
- 5 for ALPHANUMERIC
- 6 for ABBREVIATED
- 7 for RESERVED_EXTN

Attribute: RequestDeliveryReports
Scope: Server
Unit: n/a
Format: Boolean [TRUE | FALSE]
Specifies if the default behavior of the plug-in is to request delivery reports or not.

Use:

- **true** if delivery reports should be requested.
- **false** if delivery reports should not be requested.

If delivery requests are not requested, applications will, by default, not have the possibility to poll for delivery status. However it is possible to override the default setting by in the service provider SLA, application SLA, or by a custom policy rule.

**Attribute: RequestTimerValue**

Scope: Server  
Unit: milliseconds  
Format: int  

Specifies the value of the timer used when sending requests to the SMSC.

**Attribute: SMSCDefaultAlphabet**

Scope: Server  
Unit: n/a  
Format: int  

Specifies the SMSC Default Alphabet. This is the default character encoding scheme used by the SMSC when encoding short messages. The Plug-in needs to use the same character encoding scheme for the characters to be decoded correctly. All encoding schemes supported by JAVA are possible.

Use:

- **ASCII** for American Standard Code for Information Interchange  
- **Cp1252** for Windows Latin-1  
- **ISO8859_1** for ISO 8859-1, Latin alphabet No. 1.  
- **GSM_DEFAULT** for default GSM character set.

**Attribute: MobileCountryCode**

Scope: Server
Specifies the Mobile Country Code for sending operator logos.

**Attribute: MobileNetworkCode**

Scope: Server
Unit: n/a
Format: int
Specifies the Mobile Network Code for sending operator logos.

**Attribute: ReceiverProxyLocalAddressAndPort**

Scope: Server
Unit: n/a
Format: String
The address and port which the receiver of the SMPP SMSC should bind to on the server where the plug-in executes. This allows for specification of the NIC which the receiver should bind on. Default value is an empty string, which means that it will bind to the default NIC. This should be sufficient for most cases as it uses the default values, but the plug-in can also bind on specific NICs and specific local ports.

The format of this attribute is:

```
<local IP address on external NIC>:<port the SMPP SMSC should bind to on the local machine>
```

Port number must be stated, unless an empty string is used. The port number should not be set to any port number that is already bound.

Examples:

- myHostname:4767
- 10.41.26.34:7890

**Note:** You normally do not need to configure this value.

If this attribute has been changed **Operation: resetSMPPConnection** must be invoked for the changes to apply.
Attribute: SequenceNumberRangeEndId
Scope: Server
Unit: n/a
Format: int
Specifies the end ID of the Sequence Number range.

Attribute: SequenceNumberRangeStartId
Scope: Server
Unit: n/a
Format: int
Specifies the start ID of the Sequence Number range.

Attribute: SmppVersion (r)
Read only.
Displays the version of the SMPP protocol being used.

Attribute: SmscAddress
Scope: Server
Unit: n/a
Format: string
Specifies the SMSC address as an IP-address or a host name. The setting will not be applied in until the plug-in is restarted or Operation: resetSMPPConnection is performed.

Attribute: SmscPort
Scope: Server
Unit: n/a
Format: string
Specifies the port used by the SMSC. The setting will not be applied in until the plug-in is restarted or Operation: resetSMPPConnection is performed.
**Attribute: TransmitterProxyLocalAddressAndPort**

Scope: Server  
Unit: n/a  
Format: String

The address and port which the transmitter of the SMPP SMSC should bind to on the server where the plug-in executes. This allows for specification of the NIC which the transmitter should bind on.

Default value is an empty string, which means that it will bind to the default NIC. This should be sufficient for most cases as it uses the default values, but the plug-in can also bind on specific NICs and specific local ports.

The format of this attribute is:

<local IP address on external NIC>:<port the SMPP SMSC should bind to on the local machine>

Port number must be stated, unless an empty string is used. The port number should not be set to any port number that is already bound.

Examples:

- myHostname:4767
- 10.41.26.34:7890

**Note:** You normally do not need to configure this value.

If this attribute has been changed **Operation: resetSMPPConnection** must be invoked for the changes to apply.

**Attribute: UserTextMaxLength**

Scope: Server  
Unit: n/a  
Format: int

Specifies the maximum number of characters allowed in a Short Message.

**Operation: addOriginatingAddressTypeMapping**

Scope: Cluster
If a tunneled parameter, com.bea.wlcp.wlng.plugin.sms.OriginatingAddressType, is available in a request the value of the parameter is extracted and matched against the originating address type mapping list. The matching is with the parameter type.

Signature:

addOriginatingAddressTypeMapping(type: String, ton: int, npi: int)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Specifies the originating address type to be mapped.</td>
</tr>
<tr>
<td>ton</td>
<td>Specifies the ESME Type Of Number (TON). Use:</td>
</tr>
<tr>
<td></td>
<td>• 0 for UNKNOWN</td>
</tr>
<tr>
<td></td>
<td>• 1 for INTERNATIONAL</td>
</tr>
<tr>
<td></td>
<td>• 2 for NATIONAL</td>
</tr>
<tr>
<td></td>
<td>• 3 for NETWORK</td>
</tr>
<tr>
<td></td>
<td>• 4 for SUBSCRIBER</td>
</tr>
<tr>
<td></td>
<td>• 5 for ALPHANUMERIC</td>
</tr>
<tr>
<td></td>
<td>• 6 for ABBREVIATED</td>
</tr>
<tr>
<td>npi</td>
<td>Specifies the ESME Numbering Plan Indicator (NPI). Use:</td>
</tr>
<tr>
<td></td>
<td>• 0 for Unknown</td>
</tr>
<tr>
<td></td>
<td>• 1 for ISDN (E163/E164)</td>
</tr>
<tr>
<td></td>
<td>• 3 for Data (X.121)</td>
</tr>
<tr>
<td></td>
<td>• 4 for Telex (F.69)</td>
</tr>
<tr>
<td></td>
<td>• 6 for Land Mobile (E.212)</td>
</tr>
<tr>
<td></td>
<td>• 8 for National</td>
</tr>
<tr>
<td></td>
<td>• 9 for Private</td>
</tr>
<tr>
<td></td>
<td>• 10 for ERMES</td>
</tr>
<tr>
<td></td>
<td>• 14 for Internet (IP)</td>
</tr>
<tr>
<td></td>
<td>• 18 for WAP Client ID</td>
</tr>
</tbody>
</table>
Operation: **addShortCodeTranslation**

Scope: Cluster

Translates the destination address of a network-triggered SMS to a translated destination address. The translated destination address is used when matching the SMS with notification registrations. This makes it possible for an application to register for notification from a single shortcode and receive messages sent to a range of short codes (destination addresses).

**Note:** Short code translation takes place before mapping a mobile originated SMS to registered application notifications. If two or more short code translations match an original destination number the best match, the most specific, is be chosen.

**Signature:**

```java
addShortCodeTranslation(shortCodeTranslation.originalDestinationAddressPattern: String, shortCodeTranslation.translatedDestinationAddress: String)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shortCodeTranslation.originalDestinationAddressPattern</td>
<td>Pattern that is matched against the original destination address as of a SMS received from the SMSC. Regular expression. Addresses must be specified with the prefix “tel:”, for example: .^tel:6.*.</td>
</tr>
<tr>
<td>shortCodeTranslation.translatedDestinationAddress</td>
<td>The resulting translated address. Addresses must be specified with the prefix “tel:”.</td>
</tr>
</tbody>
</table>

Operation: **removeOriginatingAddressTypeMapping**

Scope: Cluster

Removes an existing TON/NPI address type mapping for a given originating address type.

**Signature:**

```java
removeOriginatingAddressTypeMapping(type: String)
```
Operation: translateOriginatingAddressNpi
Internal operation.

Operation: translateOriginatingAddressTon
Internal operation.

Operation: countOfflineNotificationCache
Read only.
Scope: Cluster
Displays the number of entries in the off-line notification cache.
Signature:
countOfflineNotificationCache()

Table 17-4  removeOnlineNotificationInfo
removeOriginatingAddressTypeMapping

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Originating address type for the mapping, see Operation: addOriginatingAddressTypeMapping and Operation: listOriginatingAddressTypeMappings.</td>
</tr>
</tbody>
</table>

Table 17-5  countOfflineNotificationCache

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>countOfflineNotificationCache</td>
<td></td>
</tr>
</tbody>
</table>

Operation: countOnlineNotificationCache
Read only.
Managing and Configuring Short Messaging Traffic Paths

Scope: Cluster
Displays the number of entries in the on-line notification cache.
Signature:
\texttt{countOnlineNotificationCache()}

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>

Table 17-6 countOnlineNotificationCache

Operation: \texttt{countSmsCache}
Read only.
Scope: Cluster
Displays the sum of short messages in the cache for mobile originating messages and mobile terminated short messages. There are separate caches (stores) for mobile originating and mobile terminating short messages but this method returns the sum.
Signature:
\texttt{countSmsCache()}

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>

Table 17-7 countSmsCache

Operation: enableReceiveSms
Read only.
Scope: Cluster
Adds an offline notification for applications that poll for mobile originated short messages. Mobile originating Short Messages matching the criteria will not result in a notification callback to an application. Instead the message is stored in Network Gatekeeper. The application has to use the correlator returned by this method to poll for Short Messages.

Signature:

\[
\text{enableReceiveSms}(\text{smsServiceActivationNumber}: \text{String}, \text{criteria}: \text{String}, \text{applicationInstanceGroupID}: \text{String})
\]

**Table 17-8  countSmsCache**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smsServiceActivationNumber</td>
<td>Destination address of the short message.</td>
</tr>
<tr>
<td>criteria</td>
<td>Text to match against to determine if the application should receive the notification.</td>
</tr>
<tr>
<td>applicationInstanceGroupID</td>
<td>Application instance group ID for the application.</td>
</tr>
</tbody>
</table>

**Operation: getOfflineNotificationInfo**

Read only.

Scope: Cluster

Displays information about a notification registered off-line, see **Operation: enableReceiveSms**.

Signature:

\[
\text{getOfflineNotificationInfo}(\text{correlator}: \text{String})
\]
Managing and Configuring Short Messaging Traffic Paths

Table 17-9  getOfflineNotificationInfo

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>Correlator identifying the notification.</td>
</tr>
</tbody>
</table>

**Operation: getOnlineNotificationInfo**

Read only.
Scope: Cluster
Displays information about a notification registered by an application
Signature:

```java
getOnlineNotificationInfo(correlator: String)
```

Table 17-10  getOnlineNotificationInfo

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>Correlator identifying the notification.</td>
</tr>
</tbody>
</table>

**Operation: listOfflineNotificationInfo**

Read only.
Scope: Cluster
Displays a list of all notifications registered off-line.
Signature:

```java
listOfflineNotificationInfo()
```
Operation: listOnlineNotificationInfo
Read only.
Scope: Cluster
Displays a list of all notifications registered by an application.
Signature:
listOnlineNotificationInfo()

Operation: listOriginatingAddressTypeMappings
Read only.
Scope: Cluster
Displays a list of all originating address type mappings.
Signature:
listOriginatingAddressTypeMappings()
Table 17-13 listOriginatingAddressTypeMappings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: listShortCodeTranslations**

Read only.

Scope: Cluster

Displays a list of all short code translations.

Signature:

`listShortCodeTranslations()`

Table 17-14 listShortCodeTranslations

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: removeOfflineNotificationInfo**

Scope: Cluster

Removes a notification registered off-line.

Signature:

`removeOfflineNotificationInfo(correlator: String)`
**Operation: removeOfflineNotificationInfo**

Scope: Cluster

Removes a notification registered by an application.

Signature:

```
removeOfflineNotificationInfo(correlator: String)
```

---

**Table 17-15 removeOfflineNotificationInfo**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>Correlator identifying the notification.</td>
</tr>
</tbody>
</table>

---

**Operation: removeOnlineNotificationInfo**

Scope: Cluster

Removes a notification registered by an application.

Signature:

```
removeOnlineNotificationInfo(correlator: String)
```

---

**Table 17-16 removeOnlineNotificationInfo**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>Correlator identifying the notification.</td>
</tr>
</tbody>
</table>

---

**Operation: removeShortCodeTranslation**

Scope: Cluster

Removes a previously added short code translation.

Signature:

```
removeShortCodeTranslation(shortCodeTranslation.originalDestinationAddress: String, shortCodeTranslation.translatedDestinationAddress: String)
```
Managing and Configuring Short Messaging Traffic Paths

### Operation: resetSMPPConnection

**Scope:** Server

Resets the connection to the SMSC. If the plug-in has stopped trying to connect to the SMSC, invoking this method will restart the reconnect procedure.

**Signature:**

`resetSMPPConnection()`

### Table 17-17  removeShortCodeTranslation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>shortCodeTranslation.origi...</code></td>
<td>Pattern that is matched against the original destination address of an SMS</td>
</tr>
<tr>
<td></td>
<td>received from the SMSC.</td>
</tr>
<tr>
<td></td>
<td>Regular expression.</td>
</tr>
<tr>
<td><code>shortCodeTranslation.trans...</code></td>
<td>The resulting translated address.</td>
</tr>
</tbody>
</table>

### Operation: startSmsNotification

**Scope:** Cluster

Registers a notification for mobile originating short messages on behalf of an application. Same as if the application used StartSmsNotification in the Parlay X 2.1 interface SmsNotificationManager.
Signature:

```java
startSmsNotification(endpoint: String, smsServiceActivationNumber: String,
criteria: String, applicationInstanceGroupID : String)
```

### Table 17-19 startSmsNotification

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint</td>
<td>Notification endpoint implemented by the application. This endpoint implements the Parlay X 2.1 Interface: SmsNotification. Format: URL.</td>
</tr>
<tr>
<td>smsServiceActivationNumber</td>
<td>Destination address to the short message address.</td>
</tr>
<tr>
<td>criteria</td>
<td>Text in the payload of the Short Message to match to determine the application to receive the notification.</td>
</tr>
<tr>
<td>applicationInstanceGroupId</td>
<td>ID of application instance group (the account username) that the application belongs to.</td>
</tr>
</tbody>
</table>

### OSA Multimedia Messaging plug-in for Parlay X 2.1 Short Messaging

Managed object: Plugin_sms_parlay_mmm

For a configuration workflow, see Configuration workflow.

### Configuration workflow

Below is an outline for configuring the plug-in using the Network Gatekeeper Management Console:

1. In Plugin_sms_parlay_mmm, configure the behavior of the plug-in by defining the following attributes. The following attributes are available:
   - Attribute: RequestTimeout
   - Attribute: MoDecodingCharset
2. Setup the routing rules to the plug-in, see Administration of plug-in routes. Use the plug-in ID Plugin_sms_parlay_mmm

3. If desired, create and load a node SLA, see Updating and loading a global node SLA. Continue with provisioning of service provider accounts and application accounts.

**Provisioning workflow**

Below is an outline for provisioning the plug-in with service provider application specific data using Network Gatekeeper Management Console:

1. In Plugin_sms_parlay_mmm, provision the following data:
   - Operation: enableReceiveSms
   - Operation: startSmsNotification

**Management operations**

The following operations are related to management:

- Operation: addShortCodeTranslation
- Operation: countOfflineNotificationCache
- Operation: countOnlineNotificationCache
- Operation: getOfflineNotificationInfo
- Operation: getOnlineNotificationInfo
- Operation: listOfflineNotificationInfo
- Operation: listOnlineNotificationInfo
- Operation: removeOfflineNotificationInfo
- Operation: removeOnlineNotificationInfo
- Operation: removeShortCodeTranslation

**Reference: attributes and operations for Plugin_sms_parlay_mmm**

Below is a list of attributes and operations for configuration and maintenance:
**Attribute: RequestTimeout**
Scope: Server
Unit: milliseconds
Format: int
Specifies the request time-out value used when sending requests to the Parlay Gateway.

**Attribute: MobileCountryCode**
Scope: Server
Unit: n/a
Format: int
Specifies the Mobile Country Code for sending operator logos.

**Attribute: MobileNetworkCode**
Scope: Server
Unit: n/a
Format: int
Specifies the Mobile Network Code for sending operator logos.

**Attribute: MoDecodingCharset**
Scope: Server
Unit:
Format: String
Specifies the character set used when decoding mobile originated Short Messages.

**Operation: addShortCodeTranslation**
Scope: Cluster
Translates the destination address of a network-triggered SMS to a translated destination address. The translated destination address is used when matching the SMS with notification registrations. This makes it possible for an application to register for notification from a single short code and receive messages sent to a range of short codes (destination addresses).
Note: Short code translation takes place before mapping a mobile originated SMS to registered application notifications. If two or more short code translations match an original destination number the best match, the most specific, is be chosen.

Signature:


Table 17-20 ShortCodeTranslation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shortCode_translation.originalDestinationAddressPattern</td>
<td>Pattern that is matched against the original destination address as of a SMS received from the Parlay Gateway. Regular expression. Addresses must be specified with the prefix “tel:”, for example: .^tel:6.*.</td>
</tr>
<tr>
<td>shortCode_translation.translatedDestinationAddress</td>
<td>The resulting translated address. Addresses must be specified with the prefix “tel:”.</td>
</tr>
</tbody>
</table>

Operation: countOfflineNotificationCache

Read only.
Scope: Cluster
Displays the number of entries in the off-line notification cache.

Signature:

countOfflineNotificationCache()
**Operation: countOnlineNotificationCache**

Read only.
Scope: Cluster
Displays the number of entries in the on-line notification cache.
Signature:

countOnlineNotificationCache()

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: enableReceiveSms**

Read only.
Scope: Cluster
 Adds an offline notification for applications that will poll for mobile originated short messages. Mobile originating Short Messages matching the criteria will not result in a notification callback to an application. Instead the message is stored in Network Gatekeeper. The application has to use the correlator returned by this method to poll for Short Messages.
Signature:

enableReceiveSms(smsServiceActivationNumber: String, criteria: String, applicationInstanceGroupID: String)
Operation: getOfflineNotificationInfo
Read only.
Scope: Cluster
Displays information about a notification registered off-line, see Operation: enableReceiveSms.
Signature:
getOfflineNotificationInfo(correlator: String)

Table 17-23 countSmsCache

<table>
<thead>
<tr>
<th>countSmsCache</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>smsServiceActivationNumber</td>
<td>Destination address of the short message.</td>
</tr>
<tr>
<td>criteria</td>
<td>Text to match against to determine the application to receive the notification.</td>
</tr>
<tr>
<td>applicationInstanceGroupId</td>
<td>Application instance group ID (the account username) for the application.</td>
</tr>
</tbody>
</table>

Operation: getOnlineNotificationInfo
Read only.
Scope: Cluster
Displays information about a notification registered by an application

Table 17-24 getOfflineNotificationInfo

<table>
<thead>
<tr>
<th>getOfflineNotificationInfo</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>correlator</td>
<td>Correlator identifying the notification.</td>
</tr>
</tbody>
</table>
getOnlineNotificationInfo(correlator: String)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>Correlator identifying the notification.</td>
</tr>
</tbody>
</table>

**Operation: listOfflineNotificationInfo**

Read only.
Scope: Cluster
Displays a list of all notifications registered off-line.
Signature:
listOfflineNotificationInfo()

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Operation: listOnlineNotificationInfo**

Read only.
Scope: Cluster
Displays a list of all notifications registered by an application.
Signature:
listOnlineNotificationInfo()
Managing and Configuring Short Messaging Traffic Paths

Table 17-27  listOnlineNotificationInfo

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Operation: listShortCodeTranslations**

Read only.

Scope: Cluster

Displays a list of all short code translations.

Signature:

listShortCodeTranslations()

Table 17-28  listShortCodeTranslations

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Operation: removeOfflineNotificationInfo**

Scope: Cluster

Removes a notification that was registered off-line.

Signature:

removeOfflineNotificationInfo(correlator: String)
Table 17-29  removeOfflineNotificationInfo

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>Correlator identifying the notification.</td>
</tr>
</tbody>
</table>

**Operation: removeOfflineNotificationInfo**

Scope: Cluster

Removes a notification registered by an application.

Signature:

\[
\text{removeOfflineNotificationInfo}(\text{correlator}: \text{String})
\]

Table 17-30  removeOnlineNotificationInfo

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>Correlator identifying the notification.</td>
</tr>
</tbody>
</table>

**Operation: removeShortCodeTranslation**

Scope: Cluster

Removes a previously added short code translation.

Signature:

\[
\text{removeShortCodeTranslation}(\text{shortCodeTranslation.originalDestinationAddress} \\
\text{Pattern}: \text{String}, \text{shortCodeTranslation.translatedDestinationAddress}: \text{String})
\]
Managing and Configuring Short Messaging Traffic Paths

---

**Table 17-31 removeShortCodeTranslation**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shortCodeTranslation.originalDestinationAddressPattern</td>
<td>Pattern that is matched against the original destination address of an SMS received from the Parlay Gateway. Regular expression.</td>
</tr>
<tr>
<td>shortCodeTranslation.translatedDestinationAddress</td>
<td>The resulting translated address.</td>
</tr>
</tbody>
</table>

**Operation: startSmsNotification**

Scope: Cluster

Registers a notification for mobile originating short messages on behalf of an application. Produces the same results as the application using StartSmsNotification in the Parlay X 2.1 interface SmsNotificationManager.

**Signature:**

```java
startSmsNotification(endpoint: String, smsServiceActivationNumber: String, criteria: String, applicationInstanceGroupID: String)
```

---

**Table 17-32 startSmsNotification**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint</td>
<td>Notification endpoint implemented by the application. This endpoint implements the Parlay X 2.1 Interface: SmsNotification. Format: URL.</td>
</tr>
<tr>
<td>smsServiceActivationNumber</td>
<td>Destination address of the short message address.</td>
</tr>
</tbody>
</table>
Table 17-32  startSmsNotification

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>criteria</td>
<td>Text in the payload of the short message to match to determine the application to receive the notification.</td>
</tr>
<tr>
<td>applicationInstanceGroupID</td>
<td>ID of application instance group (the account username) that the application belongs to.</td>
</tr>
</tbody>
</table>
CHAPTER 18

Managing and Configuring Multimedia Messaging Traffic Paths

The following sections describe configuration and maintenance attributes and operations for the Parlay X 2.1 Multimedia Messaging traffic paths. They also provides a workflow for the configuration:

- Mailboxes
- Overall configuration workflow
- Stateless adapter for the backwards compatible Multimedia Messaging traffic path
- Service capability for the backwards compatible Multimedia Messaging traffic paths
- OSA MultiMedia Messaging plug-in for the backwards compatible Multimedia Messaging traffic path
- MM7 plug-in for the backwards compatible Multimedia Messaging traffic path

Backwards Compatible Multimedia Messaging Traffic Paths

Mailboxes

All messages, both application initiated and network triggered, flowing through the Multimedia Messaging traffic paths are stored in mailboxes. Mailboxes are created and assigned to service provider and application accounts as a part of the flow, but there are some settings related to mailboxes that span all service provider and application accounts:
Managing and Configuring Multimedia Messaging Traffic Paths

- Periodic rinse attributes:
  - **Operation: setStorageDuration** specifies how long time a message should be stored in the mailbox.
  - **Attribute: RemovalTime** specifies at which time of day old messages should be removed from the mailbox.
  - **Operation: setSelectUnreadMessagesSqlSize** specifies the number of messages to be deleted in one single batch delete process, and the time to wait between individual batch delete processes. This setting is used to optimize database operation. The batch delete process is also referred to as periodic rinse.

- **Attribute: DeliveryNotificationUsage** specifies if individual application-initiated messages should be removed directly when sent, or if they should be removed when they have been marked as delivered by the network node and the delivery report has been distributed to the application that initiated the send message request. Used to increase performance.

- **Operation: setDeleteMessageBatchParameters** specifies how individual messages that are not relevant to or already have been processed by the application are deleted from the database.

The multimedia traffic paths use two common software modules for all traffic on the network tier, a stateless adapter and a service capability. When a request has passed these two layers, the plug-in selection is performed and the request is handed off to the network protocol plug-in. There are configuration settings for both of these layers.

**Overall configuration workflow**

Below is an outline for configuring the Multimedia Messaging traffic paths using the Network Gatekeeper Management Console:

1. Setup routing to the backwards compatible Multimedia Messaging traffic paths. See Administration of plug-in routes. Use the plug-in ID Plugin_mmm_BC.

2. Decide the behavior when it comes to periodic rinse, delivery notifications, and batch deletion of messages. See Mailboxes for a description. The following configuration parameters are available:
   - **Attribute: DeliveryNotificationUsage** in the managed object ESPA_messaging.
   - **Attribute: DeleteMessageMaxBatchSize**
   - **Attribute: MailboxTimeout**
   - **Attribute: MaxPropertyFetchSize**
Backwards Compatible Multimedia Messaging Traffic Paths

3. Define which network protocol plug-ins to use and follow the configuration flow for each plug-in:
   - OSA MultiMedia Messaging plug-in for the backwards compatible Multimedia Messaging traffic path
   - MM7 plug-in for the backwards compatible Multimedia Messaging traffic path

Move on to the provisioning of service provider accounts and application accounts.

Stateless adapter for the backwards compatible Multimedia Messaging traffic path

Managed object is SESPA_px_messaging and SESPA_messaging.

Reference: attributes and operations for SESPA_px_messaging

Below is a list of attributes and operations for configuration and maintenance:

- Attribute: MailboxesWithEnabledNotifications (r)
- Operation: disableAllDeliveryreportNotifications
- Operation: getDeliveryReportNotificationsForMailbox
- Operation: getMessageArrivedNotificationsForMailbox
Attribute: MailboxesWithEnabledNotifications (r)
Read-only.
Lists IDs of all mailboxes that have notifications enabled.

Operation: disableAllDeliveryreportNotifications
Disables all delivery report notifications for a specific mailbox.
Scope: Server
Signature:
   disableAllDeliveryreportNotifications(mailbox: String)

Table 18-1  disableAllDeliveryreportNotifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlator</td>
<td>The ID of the message notification provided when the notification was started.</td>
</tr>
</tbody>
</table>

Operation: getDeliveryReportNotificationsForMailbox
Lists information about all delivery report notifications enabled for a specific mailbox.
Scope: Server
Signature:
   getDeliveryReportNotificationsForMailbox(mailboxId: String)

Table 18-2  getDeliveryReportNotificationsForMailbox

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mailboxId</td>
<td>The mailbox ID to list notifications for.</td>
</tr>
</tbody>
</table>
**Operation: getMessageArrivedNotificationsForMailbox**

Lists information about all message-arrived notifications that are enabled for a specific mailbox

Scope: Server

Signature:

```
getMessageArrivedNotificationsForMailbox(mailboxId: String)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mailboxId</td>
<td>The mailbox ID to list message-arrived notifications for.</td>
</tr>
</tbody>
</table>

**Reference: attributes and operations for SESPA_messaging**

Below is a list of attributes and operations for configuration and maintenance:

- **Operation: listPerformanceEventsEnabled**
- **Operation: setPerformanceEventEnabled**

**Operation: listPerformanceEventsEnabled**

Lists IDs of all performance events and if they are enabled or not. See **Operation: setPerformanceEventEnabled** for a description of the IDs.

Scope: Server

Signature:

```
listPerformanceEventsEnabled()
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
**Operation: setPerformanceEventEnabled**

Specifies if performance events should be sent or not.

Scope: Server

Signature:

```
setPerformanceEventEnabled(eventIdentifier: int, status: boolean)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventIdentifier</td>
<td>The ID of the performance event. The following identifiers are defined:</td>
</tr>
<tr>
<td></td>
<td>2300 - sendSMS received from client</td>
</tr>
<tr>
<td></td>
<td>12301 - before sending sendSMS to ESPA</td>
</tr>
<tr>
<td></td>
<td>12302 - after sending sendSMS to ESPA</td>
</tr>
<tr>
<td></td>
<td>12303 - sendMMS received from client</td>
</tr>
<tr>
<td></td>
<td>12304 - before sending sendMMS to ESPA</td>
</tr>
<tr>
<td></td>
<td>12305 - after sending sendMMS to ESPA</td>
</tr>
<tr>
<td></td>
<td>12306 - before sending SMS notification to listener</td>
</tr>
<tr>
<td></td>
<td>12307 - after sending SMS notification to listener</td>
</tr>
<tr>
<td></td>
<td>12308 - before sending MMS notification to listener</td>
</tr>
<tr>
<td></td>
<td>12309 - after sending MMS notification to listener</td>
</tr>
<tr>
<td>status</td>
<td>Set to true if event is enabled.</td>
</tr>
</tbody>
</table>

**Service capability for the backwards compatible Multimedia Messaging traffic paths**

Managed object is ESPA_messaging

**Reference: attributes and operations for ESPA_messaging**

Below is a list of attributes and operations for configuration and maintenance:

- Attribute: DbCleanupParams (r)
- Attribute: DeleteMessageMaxBatchSize
- Attribute: DeleteMessageTimerPeriod (r)
- Attribute: DeliveryNotificationUsage
- Attribute: MailboxTimeout
- Attribute: MaxPropertyFetchSize
- Attribute: NotificationCallbacksTimeBeforeDead
- Attribute: OverloadPercentage
- Attribute: PerformanceEventsEnabled (r)
- Attribute: RemovalTime
- Attribute: ReqCounterIncrementUsage
- Attribute: SevereOverloadPercentage
- Operation: setDbCleanupParams
- Operation: setDeleteMessageBatchParameters
- Operation: setPerformanceEventEnabled
- Operation: setRemovalSqlSize
- Operation: setStorageDuration
- Operation: setSelectUnreadMessagesSqlSize
- Operation: closeMailbox
- Operation: deleteNotification
- Operation: emptyMailbox
- Operation: listNotifications

**Attribute: DbCleanupParams (r)**

Read only.
Displays the DB cleanup parameters.
**Attribute: DeleteMessageMaxBatchSize**

Read-only.
Displays message delete batch maximum batch size.

**Attribute: DeleteMessageTimerPeriod (r)**

Read-only.
Displays message delete timer period between batch deletes if there are entries in a batch that have not been executed since the last timer execution.

**Attribute: DeliveryNotificationUsage**

Scope: Cluster
Unit: n/a
Format: Boolean [true | false]
Defines if delivery notifications of messages should be stored in the database. If applications need to query the delivery status of sent messages, delivery notifications must be set to True.
Default value is True.

**Note:** Setting this parameter to false will increase the performance but will also disable the storage of delivery notifications for all applications using the Multimedia Messaging traffic paths.

**Attribute: MailboxTimeout**

Scope: Cluster
Unit: seconds
Format: int [60...500000]
Specifies the time-out value for opened mailboxes. The time-out value is specified in seconds. When a mailbox has not been used for this amount of time it is closed.

**Attribute: MaxPropertyFetchSize**

Scope: Cluster
Unit: n/a
Format: int
Defines the maximum number of message properties to fetch for a message. If the maximum number of properties is larger than the number of properties for a message, all properties will not be fetched.

**Note:** Each destination address in a send list counts as one property.

**Attribute: NotificationCallbacksTimeBeforeDead**

Scope: Cluster
Unit: seconds
Format: int

Specifies how long notification callback-interfaces should be classified as 'zombies' (not responding to heartbeat supervision) before being considered dead and removed from the notification.

If a notification only has one callback-interface set and this is considered dead the notification will be discarded.

**Note:** Callback interfaces in already enabled notifications will not be supervised by this new value until Network Gatekeeper is restarted.

**Attribute: OverloadPercentage**

Scope: Server
Unit: n/a
Format: int [0...100]

Specifies the load percentage defining when the software module will raise an overloaded alarm. Must be less than **Attribute: SevereOverloadPercentage**.

**Attribute: PerformanceEventsEnabled (r)**

Scope: Cluster
Read only.

Lists which performance events should be enabled and disabled. Performance events are predefined EDRs emitted at certain points in the execution flow. Used for measuring latency.

**Attribute: RemovalTime**

Scope: Cluster
Unit:
Format: String [HH:MM], where HH is [00...23] and MM is [0...59]
Specifies the time of day when removal of old messages will be performed.

**Attribute: ReqCounterIncrementUsage**
Scope: Cluster
Unit: n/a
Format: Boolean [TRUE | FALSE]
Defines how a send list is treated by the SLA enforcement mechanism. Either a message with several destination addresses can be treated as one request or it can be treated as multiple requests, equivalent to the number of destination addresses.

Use:
- TRUE, to increase request counters with one unit, regardless of the number of destination addresses for a message.
- FALSE, to increase request counters with a number equal to the number of destination addresses for a message.

**Attribute: SevereOverloadPercentage**
Scope: Server
Unit: n/a
Format: int [0...100]
Specifies the load percentage at which the software module will raise a severely overloaded alarm.

Must be larger than **Attribute: OverloadPercentage**.

**Operation: setDbCleanupParams**
Defines how messages are deleted from the database.

**Note:** Only applicable when using MySQL.

\`
setDbCleanupParams(params.cleanupMethod: int, params.cleanupWaitTime: int)
\`
Operation: setDeleteMessageBatchParameters

Defines how batch deletion of messages is handled. The setting is used in two different scenarios:

- When application-initiated messages have been handed off to the network and delivery notifications are disabled, see Attribute: DeliveryNotificationUsage.
- When network-triggered messages have successfully been handed off to an application via a call to the Parlay X 2.1 Multimedia Messaging MessageNotification interface.

In both cases, messages are not deleted from the mailbox immediately. Instead the messages are tagged and removed periodically in batches.

**Signature:**

```plaintext
setDeleteMessageBatchParameters(maxBatchSize: int, batchDeleteTimerPeriod: int)
```
Operation: setPerformanceEventEnabled

Defines if a performance event should be enabled or not. You should enable events only when measuring performance.

Events are sent using the SLEE Event service.

Signature:

setPerformanceEventEnabled(eventIdentifier:int, status: boolean)

Table 18-7 setDeleteMessageBatchParameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxBatchSize</td>
<td>Number of messages to be deleted per SQL query batch execution for deleting messages that are not relevant to store, because all information about the message has been delivered to the initiating or target application.</td>
</tr>
<tr>
<td>batchDeleteTimerPeriod</td>
<td>Time period between batch deletes if there are entries in a batch that have not been executed since last timer execution. Given in seconds.</td>
</tr>
</tbody>
</table>

Operation: setRemovalSqlSize

Deprecated and unused.

Specifies the number of messages to be deleted per SQL query batch execution.
Signature:

```java
setRemovalSqlSize(messages: int)
```

<table>
<thead>
<tr>
<th>Table 18-9  setRemovalSqlSize</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>setRemovalSqlSize</strong></td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
</tr>
<tr>
<td>messages</td>
</tr>
</tbody>
</table>

**Operation: setSelectUnreadMessagesSqlSize**

Deprecated and unused.

Specifies the number of unread messages to be selected per SQL query execution.

Signature:

```java
setSelectUnreadMessagesSqlSize(messages: int)
```

<table>
<thead>
<tr>
<th>Table 18-10  setRemovalSqlSize</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>setRemovalSqlSize</strong></td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
</tr>
<tr>
<td>messages</td>
</tr>
</tbody>
</table>

**Operation: setStorageDuration**

Deprecated and unused.

Specifies the number of days a message will be stored in the mailbox before it is automatically removed.

Signature:

```java
setStorageDuration(days: int)
```
Managing and Configuring Multimedia Messaging Traffic Paths

Table 18-11  setStorageDuration

<table>
<thead>
<tr>
<th>setStorageDuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>days</td>
</tr>
</tbody>
</table>

**Operation: closeMailbox**

Closes the specified mailbox.

Signature:

`closeMailbox(mailboxAddress: String)`

Table 18-12  closeMailbox

<table>
<thead>
<tr>
<th>closeMailbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>mailboxAddress</td>
</tr>
</tbody>
</table>

**Operation: deleteNotification**

Deletes a notification.

Signature:

`deleteNotification(id: int)`

Table 18-13  deleteNotification

<table>
<thead>
<tr>
<th>deleteNotification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>id</td>
</tr>
</tbody>
</table>
**Operation: emptyMailbox**

Removes all messages in a specific mailbox.

**Signature:**

emptyMailbox(addr: String)

<table>
<thead>
<tr>
<th>Table 18-14 emptyMailbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>emptyMailbox</td>
</tr>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>addr</td>
</tr>
</tbody>
</table>

**Operation: listNotifications**

Lists enabled notifications, with details about each notification, for a given service provider account and application account.

listNotifications(applicationId: String, serviceProviderId: String, startIndex: int, maxCount: int)

<table>
<thead>
<tr>
<th>Table 18-15 listNotifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>listNotifications</td>
</tr>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>applicationId</td>
</tr>
<tr>
<td>serviceProviderId</td>
</tr>
<tr>
<td>startIndex</td>
</tr>
<tr>
<td>maxCount</td>
</tr>
</tbody>
</table>
OSA MultiMedia Messaging plug-in for the backwards compatible Multimedia Messaging traffic path

Managed object: Plugin_OSA_MM_MMS

For a configuration workflow, see Configuration workflow.

Configuration workflow

Below is an outline for configuring the plug-in:

1. Gather information about the OSA Gateway and configure the plug-in accordingly. The following information needs to be obtained from the OSA Gateway administrator and configured in the OSA Access service:
   - OSA/Parlay SCS type to be used in the look up (service discovery) phase when requesting the service (OSA/Parlay SCS) from the OSA/Parlay Gateway. Typically this is P_MULTIMEDIA_MESSAGING.
   - OSA/Parlay service properties to be used in the look up (service discovery) phase when requesting a service (OSA/Parlay SCS) from the OSA/Parlay Gateway. This depends on the OSA Gateway implementation. Typically this is property TpMessageDeliveryType and value P_MMM_MMS.
   - Authentication type used by the OSA/Parlay Framework.
   - Encryption method used for the connection with the OSA Gateway.
   - Signing algorithm used when signing the service level agreement with the OSA/Parlay Framework.

2. Setup the OSA Client and the OSA Client Mappings according to Creating an OSA client and Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS in.

3. In Plugin_OSA_MM_MMS, configure the plug-in:
   - Attribute: MaxNumberOfMtRetries
   - Attribute: MoDecodingCharset
   - Attribute: MTRetryInterval
   - Attribute: NotificationRestoreInterval
   - Operation: addScsProperty
   - Operation: disableNotificationRestoration or Operation: enableNotificationRestoration
4. Setup the routing rules to the plug-ins, see Administration of plug-in routes. The plug-in ID to use is generated, in numeric form. The ID is presented next to the value Plugin_OSA_MM_MMS, see Operation: listPluginTypes.

5. If desired, create and load a node SLA, see Updating and loading a global node SLA.

**Reference: attributes and operations for Plugin_OSA_MM_MMS**

Below is a list of attributes and operations for configuration and maintenance.

- **Attribute: MaxNumberOfMtRetries**
- **Attribute: MoDecodingCharset**
- **Attribute: MTRetryInterval**
- **Attribute: NotificationRestoreInterval**
- **Operation: addScsProperty**
- **Operation: deleteScsProperty**
- **Operation: disableNotificationRestoration**
- **Operation: enableNotificationRestoration**
- **Operation: getNotification**
- **Operation: listNotifications**
- **Operation: listScsProperties**

**Attribute: MaxNumberOfMtRetries**

Scope: Cluster

Unit: n/a

Format: int

Specifies the maximum number of times a mobile terminated message should be resent to the OSA/Parlay Gateway in case the message send operation fails.

**Attribute: MoDecodingCharset**

Scope: Cluster

Unit: n/a
Managing and Configuring Multimedia Messaging Traffic Paths

Format: String
Specifies the default character set used by the OSA Gateway for mobile originated MMSes. Enter the encoding scheme. It must be supported by Java: for example:
- ASCII
- Cp1252
- ISO8859_1
See http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html

Attribute: MTRetryInterval
Scope: Cluster
Unit: seconds
Format: int
Specifies the time between each retry of sending failed mobile terminated messages to the OSA Gateway.

Attribute: NotificationRestoreInterval
Scope: Cluster
Unit: seconds
Format: String
Specifies the time between each notification restoration process. See Operation: enableNotificationRestoration.

Operation: addScsProperty
Adds an OSA service property. OSA service properties are used in the lookup (service discovery) phase when requesting a service (OSA SCS) from the OSA Gateway. Duplicate name-value pair combinations cannot be added, whereas a single name and multiple values are allowed.

Signature:
addScsProperty(name: String, value: String)
Table 18-16  addScsProperty

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the service property. Example service version.</td>
</tr>
<tr>
<td>value</td>
<td>The value of the property. Example 4.1</td>
</tr>
</tbody>
</table>

**Operation: deleteScsProperty**

Deletes a service property. Both name and value must match.

Signature:

```
deleteScsProperty(name: String, value: String)
```

Table 18-17  deleteScsProperty

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the service property to be deleted.</td>
</tr>
<tr>
<td>value</td>
<td>The value of the service property to be deleted.</td>
</tr>
</tbody>
</table>

**Operation: disableNotificationRestoration**

Disables automatic restoration of notifications.

Signature:

```
disableNotificationRestoration()
```
**Operation: enableNotificationRestoration**

Enables automatic restoration of notifications.

**Signature:**

```
enableNotificationRestoration()
```

**Table 18-19 enableNotificationRestoration**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Operation: disableNotification**

Disables a previously made OSA notification.

**Signature:**

```
disableNotification(id: int)
```

**Table 18-20 disableNotification**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>ID of the notification to disable.</td>
</tr>
</tbody>
</table>

**Operation: getNotification**

Returns information about specific OSA/Parlay notification.

**Signature:**

```
getNotification(id:int)
```
**Operation: listNotifications**
Displays a list of all enabled OSA/Parlay notification IDs.
Signature:
```
listNotifications()
```

**Table 18-21 getNotification**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the notification</td>
</tr>
</tbody>
</table>

**Operation: listScsProperties**
Displays a list of all service property pairs.
The format is <property name 1>/<value 1>...<property name n>/<value n>
Signature:
```
listScsProperties()
```

**Table 18-22 listNotifications**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 18-23 listScsProperties**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MM7 plug-in for the backwards compatible Multimedia Messaging traffic path

Managed object: Plugin_messaging_MM7

For a configuration workflow, see Configuration workflow.

Configuration workflow

Below is an outline for configuring the plug-in:

1. In Plugin_messaging_MM7, configure the behavior of the plug-in by defining the following attributes:
   - Attribute: AttachmentFormatType
   - Attribute: DefaultDestinationAddress
   - Attribute: HTTPBasicAuthentication
   - Attribute: MM7Version
   - Attribute: MMSRelayServerAddress
   - Attribute: MMSRelayServerPath
   - Attribute: MMSRelayServerPort
   - Attribute: MMSRelayServerURN
   - Attribute: ReadReportRequestStatus
   - Attribute: ReportAddress
   - Attribute: SequenceNumberRangeEndId
   - Attribute: SequenceNumberRangeStartId
   - Attribute: VASID
   - Attribute: VASPID
   - Operation: setDBCleanerParameters
   - Operation: setHTTPBasicAuthenticationDetails
   - Operation: setHeartbeatConfiguration
2. In Plugin_messaging_MM7, configure other attributes related to the handling of database queries and overload:
   - Attribute: OverloadPercentage
   - Attribute: QueryLimitRangeSize
   - Attribute: SevereOverloadPercentage
   - Operation: setPerformanceEventEnabled

3. Setup the routing rules for the plug-ins, see Administration of plug-in routes. The plug-in ID to use is generated, in numeric form. The ID is presented next to the value Plugin_messaging_MM7, see Operation: listPluginTypes.

4. If desired, create and load a node SLA, see Updating and loading a global node SLA. Move on to the provisioning of service provider accounts and application accounts.

Reference: attributes and operations for Plugin_messaging_MM7

Below is a list of attributes and operations for configuration and maintenance:

- Attribute: AttachmentFormatType
- Attribute: DBCleanerParameters (r)
- Attribute: DefaultDestinationAddress
- Attribute: HTTPBasicAuthentication
- Attribute: HTTPBasicAuthenticationDetails (r)
- Attribute: HeartbeatConfiguration (r)
- Attribute: MM7Version
- Attribute: MMSRelayServerAddress
- Attribute: MMSRelayServerPath
- Attribute: MMSRelayServerPort
- Attribute: MMSRelayServerURN
- Attribute: OverloadPercentage
- Attribute: PerformanceEventsEnabled (r)
Attribute: QueryLimitRangeSize
Attribute: ReadReportRequestStatus
Attribute: ReportAddress
Attribute: SequenceNumberRangeEndId
Attribute: SequenceNumberRangeStartId
Attribute: SevereOverloadPercentage
Attribute: VASID
Attribute: VASPID
Operation: setDBCleanerParameters
Operation: setHTTPBasicAuthenticationDetails
Operation: setHeartbeatConfiguration
Operation: setPerformanceEventEnabled
Attribute: NumberofOutstandingMessageReqs (r)
Operation: explicitlyActivatePlugin

Attribute: AttachmentFormatType
Scope: Server
Unit: n/a
Format: String [Dime | Mime]
Specifies the format used for attachments. One of the following:

- Dime
- Mime (default)

Attribute: DBCleanerParameters (r)
Read only.
Displays settings for the store and forward database cleaner service:
Invocation time: Time of day when the service should be invoked (hh:mm)
Invocation interval: How often (no. of days between invocations) the service should be invoked
Age of requests (hours) to be deleted

**Attribute: DefaultDestinationAddress**
Scope: Server
Unit: n/a
Format: String
Specifies a default destination address for incoming messages without a destination address. If set to an empty string, incoming messages without a destination address will be rejected.

**Attribute: HTTPBasicAuthentication**
Scope: Server
Unit: n/a
Format: Boolean [TRUE | FALSE]
Specifies if HTTP basic authentication is enabled or not.

**Attribute: HTTPBasicAuthenticationDetails (r)**
Read-only.
Displays HTTP basic authentication support details, including user name and password.

**Attribute: HeartbeatConfiguration (r)**
Read-only.
Displays the heartbeat configuration, see Operation: setHeartbeatConfiguration.

**Attribute: MM7Version**
Scope: Server
Unit: n/a
Format: String [5.3.0 | ericsson_mm7_1_0 | ericsson_mm7_R2_0_ACA_03 | ericsson_mm7_R2_5_ACA_04]
Specifies the MM7 version to be used. Enter:
Managing and Configuring Multimedia Messaging Traffic Paths

- **5.3.0** for MM7 version 5.3.0
- **ericsson_mm7_1_0** for Ericsson specific MM7 version.
- **ericsson_mm7_R2_0_ACA_03** additional Ericsson specific version
- **ericsson_mm7_R2_0_ACA_04** additional Ericsson specific version

**Attribute: MMSRelayServerAddress**
Scope: Server
Unit: n/a
Format: String
Specifies the IP-address used in the HTTP request to the MMS relay/server.

**Attribute: MMSRelayServerPath**
Scope: Server
Unit: n/a
Format: String
Specifies the path used in the HTTP request to the MMS relay/server.
For example, if the path is configured to be /mms-rs/mm7 the first line of the HTTP request will be **POST /mms-rs/mm7 HTTP/1.1**
The Ericsson specific MM7 implementation is typically found at /vasp/servlet/messagerouter

**Attribute: MMSRelayServerPort**
Scope: Server
Unit: n/a
Format: integer
Specifies the port the MMS Relay/Server is listening on.

**Attribute: MMSRelayServerURN**
Scope: Server
Unit: n/a
Format: String
Specifies the MMS relay/server URN (Uniform Resource Name) target.
For example, if the URN target is configured to be urn:mm7 the first line in the SubmitReq message is <SubmitReq xmlns="urn:mm7">
The Ericsson MM7 implementation is typically registered at urn:MM7Submit.Req

**Attribute: OverloadPercentage**
Scope: Server
Unit: n/a
Format: int [0...100]
Specifies the load percentage at which the plug-in will raise an overloaded alarm. Must be less than **Attribute: SevereOverloadPercentage**.

**Attribute: PerformanceEventsEnabled (r)**
Read-only.
Displays a list of performance events and if they are enabled for this plug-in and server or not.

**Attribute: QueryLimitRangeSize**
Scope: Server
Unit: number of entries
Format: int
Specifies the size of the database query limit range used when cleaning up old entries in the database. Other details are specified in Operation: setDBCleanerParameters.

**Attribute: ReadReportRequestStatus**
Scope: Server
Unit: n/a
Format: Boolean [TRUE | FALSE]
Specifies if read reports are requested or not. Enter:
• **TRUE** to ask for read reports.
• **FALSE** to not ask for read reports.
**Attribute: ReportAddress**
Scope: Server
Unit: n/a
Format: String
Specifies the report address to be used when sending messages to an Ericsson MMSC that requires the report address element. If set to an empty string, no `Report-Address` element will be included in the MM7 request.

**Attribute: SequenceNumberRangeEndId**
Scope: Server
Unit: n/a
Format: String
Unused parameter.

**Attribute: SequenceNumberRangeStartId**
Scope: Server
Unit: n/a
Format: String
Unused parameter.

**Attribute: SevereOverloadPercentage**
Scope: Server
Unit: n/a
Format: int [0...100]
Specifies the load percentage at which the plug-in will raise a severely overloaded alarm.
Must be larger than **Attribute: OverloadPercentage**.

**Attribute: VASID**
Scope: Server
Unit: n/a
Format: String
specifies the VAS (Value Added Service) ID to be used for the plug-in when connecting to the MMSC.
If left empty the application account ID related to each request is used.

**Attribute: VASPID**

Scope: Server
Unit: n/a
Format: String
Specifies the VASP (Value Added Service Provider) ID to be used for the plug-in when connecting to the MMSC.
If left empty, the service provider ID related to each request is used.

**Operation: setDBCleanerParameters**

Specifies parameters for the store and forward database cleaner service. This service is an internal service in the plug-in that removes old entries from the store and forward database.

Signature:

```
setDBCleanerParameters(cleanupTime: String, cleanupInterval: int, dbEntryAgeLimit: int)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cleanupTime</td>
<td>Specifies time of day when the DB cleanup service should be invoked. Given in hh:mm, where hh = [00...23] and mm = [00...59].</td>
</tr>
<tr>
<td>cleanupInterval</td>
<td>Specifies how often the DB Cleaner service should be invoked. Given in number of days between invocations.</td>
</tr>
<tr>
<td>dbEntryAgeLimit</td>
<td>Specifies the minimum age of entries to be deleted. Given in hours.</td>
</tr>
</tbody>
</table>
Operation: setHTTPBasicAuthenticationDetails

Specifies the authentication details to be used for the HTTP basic authentication towards the MMSC.

Signature:

$$\text{setHTTPBasicAuthenticationDetails( authInformation.username: String, authInformation.password: String)}$$

Table 18-25 setHTTPBasicAuthenticationDetails

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>authInformation.username</td>
<td>Specifies the user name.</td>
</tr>
<tr>
<td>authInformation.password</td>
<td>specifies the password.</td>
</tr>
</tbody>
</table>

Operation: setHeartbeatConfiguration

Specifies the plug-in to MM7 Relay/server heartbeat function configuration.

Signature:

$$\text{setHeartbeatConfiguration(hbConf.heartbeatURL: String, hbConf.heartbeatInterval: long, hbConf.heartbeatContentMatch: String, explicitlyActivatePlugin: boolean)}$$

Table 18-26 setHeartbeatConfiguration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hbConf.heartbeatURL</td>
<td>Specifies the MM7 Relay/server URL to use for the heartbeats.</td>
</tr>
<tr>
<td>hbConf.heartbeatInterval</td>
<td>Specifies the interval between heartbeats. Set the interval to 0 if the heartbeat mechanism should be disabled. Given in milliseconds.</td>
</tr>
</tbody>
</table>
Example: If the URL is set to http://192.168.1.4:8080/status/status.txt and status.txt contains the text MM7 server OK, the plug-in will match this text against the text specified in heartbeat content match. If they match, the connection is considered OK.

**Operation: setPerformanceEventEnabled**

Defines if a performance event should be enabled or not. You should enable events only when measuring performance.

Events are sent using the SLEE Event service.

**Signature:**

`setPerformanceEventEnabled(eventIdentifier: int, status: boolean)`
Attribute: NumberOfOutstandingMessageReqs (r)
Read only.
Displays the number of outstanding message requests, that is message requests that are waiting for a delivery report.

Operation: explicitlyActivatePlugin
Activates or deactivate the plug-in. This operation is useful in situations where the heartbeat has been turned off and the plug-in is in a deactivated state.

Signature:
explicitlyActivatePlugin(isActive: boolean)

Table 18-28 explicitlyActivatePlugin

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isActive</td>
<td>Use:</td>
</tr>
<tr>
<td></td>
<td>• True if the plug-in should be put in state activated.</td>
</tr>
<tr>
<td></td>
<td>• false if the plug-in should be put in state deactivated.</td>
</tr>
</tbody>
</table>
Managing and Configuring Payment Traffic Paths

The following sections describe configuration and maintenance attributes and operations for the Parlay X 2.1 Payment traffic paths. They also provide a workflow for configuration:

- Backwards compatible Payment Traffic Paths
  - Service capability for backwards compatible Payment traffic path
  - OSA Charging plug-in for Parlay X 2.1 Payment backwards compatible traffic paths

Backwards compatible Payment Traffic Paths

This traffic path is used to hand off Payment related operations to network nodes that hold charging information. It does not keep track of accounts, but relies on external systems to maintain this information.

This path uses two common software modules for all traffic on the network tier, a stateless adapter and a service capability. When a request has passed these two layers, the plug-in selection is performed and the request is handed off to the network protocol plug-in. There are configuration settings for both of these layers.

Overall configuration workflow

Below is an outline of how to configure the Payment traffic path using Network Gatekeeper Management Console:

1. Setup routing to the backwards compatible Payment traffic path. See Administration of plug-in routes. Use the plug-in ID Plugin_Payment_bc.
2. In the managed object ESPA_charging set the following configurations, if needed:
   – Attribute: OverloadPercentage
   – Attribute: RequestTimeout
   – Attribute: SevereOverloadPercentage

3. Define which network protocol plug-ins to use and follow the configuration flow for each plug-in:
   – OSA Charging plug-in for Parlay X 2.1 Payment backwards compatible traffic paths

Move on to the provisioning of service provider accounts and application accounts.

**Service capability for backwards compatible Payment traffic path**

Managed object: ESPA_charging

**Reference: attributes and operations for ESPA_charging**

Below is a list of attributes and operations for configuration and maintenance.

- Attribute: OverloadPercentage
- Attribute: RequestTimeout
- Attribute: SevereOverloadPercentage

**Attribute: OverloadPercentage**

Scope: Server
Unit: percent
Format: int [0...100]

Specifies the load percentage at which the software module will raise an overload alarm. Must be less than Attribute: SevereOverloadPercentage.

**Attribute: RequestTimeout**

Scope: Cluster
Unit: seconds
Format: int
Sets the time-out (in seconds) for the requests made on the interface of the module.

**Attribute: SevereOverloadPercentage**
Scope: Server
Unit: percent
Format: int [0...100]

Specifies the load percentage at which the software module will raise a severely overloaded alarm.

Must be larger than **Attribute: OverloadPercentage**.

**OSA Charging plug-in for Parlay X 2.1 Payment backwards compatible traffic paths**

Managed object: Plugin_OSA_Charging

For a configuration workflow, see [Configuration workflow](#).

Most of the configuration for the OSA Charging for Payment is done in the OSA Access module, but with configuration details for the OSA User Location SCS. See [Managing and Configuring OSA/Parlay Gateway Connections](#).

**Configuration workflow**

Below is an outline on how to configure the plug-in:

1. Gather information about the OSA Gateway and configure the plug-in accordingly. The following information needs to be obtained from the OSA Gateway administrator and configured in the OSA Access service:
   - OSA/Parlay SCS type to be used in the look up (service discovery) phase when requesting the service (OSA/Parlay SCS) from the OSA/Parlay Gateway. Typically this is P_CHARGING.
   - OSA/Parlay service properties to be used in the look up (service discovery) phase when requesting a service (OSA/Parlay SCS) from the OSA/Parlay Gateway. This depends on the OSA Gateway implementation.
   - Authentication type used by the OSA/Parlay Framework.
   - Encryption method used for the connection with the OSA Gateway.
– Signing algorithm used when signing the service level agreement with the OSA/Parlay Framework.

2. Setup the OSA Client and the OSA Client Mappings according to Creating an OSA client and Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS.

3. Setup the routing rules to the plug-ins, see Administration of plug-in routes. The plug-in ID to use is generated, in numeric form. The ID is presented next to the value Plugin_OSA_Charging, see Operation: listPluginTypes.

4. If desired, create and load a node SLA, see Updating and loading a global node SLA.

**Reference: attributes and operations for Plugin_OSA_Charging**

Below is a list of attributes and operations for configuration and maintenance.

- **Attribute: NumberofActiveSessions (r)**

**Attribute: NumberofActiveSessions (r)**

Read-only.

Displays the number of active sessions towards the Charging SCS.
Managing and Configuring Terminal Status Traffic Paths

The following section describes configuration and maintenance attributes and operations for the Parlay X 2.1 Terminal Status traffic path. It also provides a workflow for the configuration.

- **Terminal Status Traffic Paths**
  - OSA User Status plug-in for the Parlay X 2.1 Terminal Status traffic path

**Terminal Status Traffic Paths**

The Terminal Status traffic path comes with support for the following network plug-ins:

- **OSA User Status plug-in for the Parlay X 2.1 Terminal Status traffic path**

**OSA User Status plug-in for the Parlay X 2.1 Terminal Status traffic path**

This section contains a description of the configuration attributes and operations available for the OSA User Status plug-in for the Terminal Status traffic path.

All attributes and operations in the Network Gatekeeper part are available in the managed object `Plugin_terminal_status_parlay_us`.

For a configuration workflow, see Configuration workflow.

The plug-in ID is: `Plugin_terminal_status_parlay_us`
Configuration workflow

1. In Network Gatekeeper Management Console, use Plugin终端_status_parlay_us, and configure:
   - Attribute: RequestTimeout

Below is an outline for configuring the plug-in:

2. Gather information about the OSA Gateway and configure the plug-in accordingly. The following information needs to be obtained from the OSA Gateway administrator and configured in the OSA Access service:
   - OSA/Parlay SCS type to be used in the look up (service discovery) phase when requesting the service (OSA/Parlay SCS) from the OSA/Parlay Gateway. Typically this is P_USER_STATUS.
   - OSA/Parlay service properties to be used in the look up (service discovery) phase when requesting a service (OSA/Parlay SCS) from the OSA/Parlay Gateway. This depends on the OSA Gateway implementation.
   - Authentication type used by the OSA/Parlay Framework.
   - Encryption method used for the connection with the OSA Gateway.
   - Signing algorithm used when signing the service level agreement with the OSA/Parlay Framework.

3. Setup the OSA Client and the OSA Client Mappings according to Creating an OSA client and Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS.

4. Setup the routing rules to the plug-ins, see Administration of plug-in routes. The plug-in ID is Plugin_terminal_status_parlay_us.

5. If desired, create and load a node SLA, see Updating and loading a global node SLA.

Reference: attributes and operations for Plugin终端_status_parlay_us

Below is a list of attributes and operations for configuration and maintenance.

Attribute: RequestTimeout
Operation: listNotificationInfo
Operation: removeNotificationInfo
**Attribute: RequestTimeout**
Scope: Cluster
Unit: seconds
Format: int
Sets request time out for requests when synchronizing synchronous Parlay X operations with asynchronous Parlay/OSA operations.

**Operation: listNotificationInfo**
Scope: Cluster
Displays a list of all registered notifications.
Signature:

```plaintext
listNotificationInfo()
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: removeNotificationInfo**
Scope: Cluster
Removes a registered notification.
Signature:

```plaintext
removeNotificationInfo(p1: int)
```
### Table 20-2 removeNotificationInfo

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>ID of notification.</td>
</tr>
</tbody>
</table>
Managing and Configuring Terminal Location Traffic Paths

The following section describes configuration and maintenance attributes and operations for the Parlay X 2.1 Terminal Location traffic paths. It also provides a workflow for the configuration:

- **Backwards Compatible Terminal Location Traffic Paths**
  - Overall configuration workflow
  - Stateless adapter for backwards compatible Terminal Location traffic paths
  - Stateless adapter for backwards compatible Terminal Location traffic paths
  - MLP plug-in for backwards compatible Parlay X 2.1 Terminal Location traffic path
  - OSA User Location plug-in for backwards compatible Terminal Location traffic path

**Backwards Compatible Terminal Location Traffic Paths**

All Terminal Location related operations are handed off to network nodes capable of positioning a terminal based on its address.

These traffic paths do not themselves monitor the terminal for periodic or triggered notifications, but rely on the underlying network nodes to deliver notifications.

These paths use two common software modules for all traffic on the network tier, a stateless adapter and a service capability. When a request has passed these two layers, the plug-in selection is performed and the request is handed off to the network protocol plug-in. There are configuration settings for both of these layers.

The traffic paths support two types of requests: polling type and triggered type.
The polling type request polls for a position and returns the position synchronously; in the triggered case, subscriptions for notifications are established that send information when a set of criteria is fulfilled.

Notifications are set up using triggers. Triggers define the criteria under which a notification should be sent to the application that subscribed for the notification.

The trigger criteria are of two types:

- Periodic, based on time intervals. The position of a terminal is reported periodically.
- Geographical, based on proximity to a geographical position. The position of a terminal is reported when the terminal enters or leaves the area.

**Overall configuration workflow**

Below is an outline for configuring the Terminal Location traffic paths using Network Gatekeeper Management Console:

1. Setup routing to the backwards compatible Terminal Location traffic paths. See Administration of plug-in routes. Use the plug-in ID Plugin_terminal_location_bc.

2. Change attributes, as necessary, in the managed object ESPA_userlocation:
   - Attribute: OverloadPercentage
   - Attribute: PluginType
   - Attribute: SevereOverloadPercentage

3. Define which network protocol plug-in to use and follow the configuration flow for each plug-in:
   - MLP plug-in for backwards compatible Parlay X 2.1 Terminal Location traffic path
   - OSA User Location plug-in for backwards compatible Terminal Location traffic path

Move on to the provisioning of Service provider accounts and application accounts.

**Stateless adapter for backwards compatible Terminal Location traffic paths**

Managed object: SESPA_userlocation
Reference: attributes and operations for SESPA_userlocation

Below is a list of attributes and operations for configuration and maintenance:

- Attribute: RequestTimeout

Attribute: RequestTimeout

Scope: Server
Unit: seconds
Format: int

Specifies the time out value for operations on this software module. If an operation takes longer time than specified, the operation is considered faulty and an exception is thrown.

Service capability for backwards compatible Terminal Location traffic paths

Managed object: ESPA_userlocation

Reference: attributes and operations for ESPA_userlocation

Below is a list of attributes and operations for configuration and maintenance:

- Attribute: MaxActiveTriggerCount
- Attribute: MaximumOutstandingPeriodicAddresses
- Attribute: MaxTriggerAddresses
- Attribute: MaxTriggerDuration
- Attribute: MaxTriggerNotifications
- Attribute: MinimumPeriodicInterval
- Attribute: MinTimeBetweenTriggerNotifications
- Attribute: MinimumPeriodicInterval
- Attribute: MinTriggerAccuracy
- Attribute: OverloadPercentage
- Attribute: PluginType
Managing and Configuring Terminal Location Traffic Paths

- Attribute: ReqCounterIncrementUsage
- Attribute: RequestTimeout
- Attribute: SevereOverloadPercentage
- Attribute: TriggerGuillotineTimeout
- Attribute: TriggerGuillotineTimeout

**Attribute: MaxActiveTriggerCount**
Scope: Server
Unit: n/a
Format: int
Specifies the maximum number of concurrent active triggers for triggered user location per service provider.

**Attribute: MaximumOutstandingPeriodicAddresses**
Scope: Server
Unit: n/a
Format: int
Specifies the maximum number of location addresses (numbers) that can be registered for a specific periodic user location trigger.

**Attribute: MaxTriggerAddresses**
Scope: Server
Unit: n/a
Format: int
Specifies the maximum number of addresses that are allowed per location trigger.

**Attribute: MaxTriggerDuration**
Scope: Server
Unit: seconds
Format: int
Specifies the maximum lifetime of a location trigger.

**Attribute: MaxTriggerNotifications**

Scope: Server  
Unit: n/a  
Format: int  
Specifies the maximum number of reports per location trigger.

**Attribute: MinimumPeriodicInterval**

Scope: Server  
Unit: Seconds  
Format: int  
Specifies minimum allowed time interval between periodic location requests for periodic triggers.

**Attribute: MinTimeBetweenTriggerNotifications**

Scope: Server  
Unit: Seconds  
Format: int  
Specifies the minimum time interval between notifications.

**Attribute: MinimumPeriodicInterval**

Scope: Server  
Unit: Seconds  
Format: int  
Specifies minimum allowed time interval between periodic location requests.

**Attribute: MinTriggerAccuracy**

Scope: Server  
Unit: meters  
Format: int
Specifies the minimum accuracy for the geographical trigger area. If the requested accuracy is more granular (smaller) than this value, it will replace the value in the request from the application with this value.

**Attribute: OverloadPercentage**
Scope: Server
Unit: n/a
Format: int [0...100]
Specifies the load percentage at which the software module will raise an overload alarm. Must be less than **Attribute: SevereOverloadPercentage**.

**Attribute: PluginType**
Scope: Cluster
Unit: n/a
Format: boolean
Specifies, in case of multiple target addresses being positioned in a single request, whether a single error callback should be performed for all failed target addresses or multiple error callbacks should be performed, one for each failed target address. Enter:

- TRUE to expect one error callback for the whole request
- FALSE to expect one error callback for each failed address

**Attribute: ReqCounterIncrementUsage**
Scope: Cluster
Unit: n/a
Format: boolean
Specifies how request counters should be incremented for user location requests:

- TRUE the number of location targets should be used as increment value for request counters.
- FALSE the request counter is updated with one per request.
Attribute: RequestTimeout
Scope: Server
Unit: seconds
Format: int
Specifies the time out value for operations on this software module. If an operation takes longer time than specified, the operation is considered faulty and an exception is thrown.

Attribute: SevereOverloadPercentage
Scope: Server
Unit: n/a
Format: int [0...100]
Specifies the load percentage at which the software module will raise a severely overloaded alarm.
Must be larger than Attribute: OverloadPercentage.

Attribute: TriggerGuillotineTimeout
Scope: Server
Unit: seconds
Format: int
Specifies the maximum time a location trigger is valid. When this time period expires, no more location notifications will be sent to the application for the specific trigger.

MLP plug-in for backwards compatible Parlay X 2.1
Terminal Location traffic path
Managed object: Plugin_user_location_MLP

Mode of operation
The MLP plug-in can be configured to operate in either MLP 3.0 or MLP 3.2 mode. When operating in 3.0 mode, no application-initiated requests regarding triggered positioning are valid since the standard does not support this. From an application point-of-view, errors will be
returned when performing a request that is mapped to MLP 3.2 operations and the plug-in is configured to operate in MLP 3.0 mode.

Some configuration settings are not applicable when the plug-in operates in MLP 3.2 mode.

**Request buffering**

The MLP plug-in can be configured to use a request buffer.

If a request towards the MLP server fails, the request is temporarily stored in a buffer. At regular intervals, the plug-in tries to re-send the buffered requests. In order not to overload the MLP server at a buffer flush point, the maximum number of requests to be sent in one flush can be configured.

If request buffering is used, the following settings can be configured:

- **Buffer size**: the size of the buffer
- **Buffer burst interval**: the time interval between flushes
- **Buffer burst size**: the number of request to be sent in one flush

The setting of these attributes depends mainly on the capacity of Network Gatekeeper, the MLP server, the network connecting them and more.

The behavior towards applications is slightly different when request buffering is enabled than when it is not enabled. If request buffering is not used, the application will receive an error stating that the connection is not available immediately. If request buffering is used, this error is not reported until the maximum size of the buffer is reached.

**Configuration workflow**

Below is an outline for configuring the MLP plug-in using the Network Gatekeeper Management Console:

1. Configure the behavior of the plug-in by defining the following attributes in the managed object Plugin_user_location_MLP. The following attributes can be configured:
   - Attribute: CharacterEncoding
   - Attribute: MlpAltitudeSupported
   - Attribute: MlpDelay
   - Attribute: MlpLocationEstimatesOK
   - Attribute: MlpLocationReqType
Attribute: MlpPassword
Attribute: MlpPushAddr
Attribute: MlpRequestor
Attribute: MlpServiceId
Attribute: MlpSrsName
Attribute: MlpUrl
Attribute: MlpUser
Attribute: MlpVersionSupported
Attribute: XMLDocTypeTagUsage
Attribute: BufferFlushInterval
Attribute: BurstInterval
Attribute: BurstSize
Attribute: HACheckInterval
Attribute: OverloadPercentage
Attribute: RequestBuffering
Attribute: SevereOverloadPercentage

2. Setup the routing rules to the plug-ins, see Administration of plug-in routes. The plug-in ID to use is generated, in numeric form. The ID is presented next to the value Plugin_user_location_MLP, see Operation: listPluginTypes.

3. If desired, create and load a node SLA, see Updating and loading a global node SLA. Move on to provisioning of service provider accounts and application accounts.

**Reference: attributes and operations for Plugin_user_location_MLP**

Below is a list of attributes and operations for configuration and maintenance:

- Attribute: BufferFlushInterval
- Attribute: BurstInterval
- Attribute: BurstSize
- Attribute: CharacterEncoding
- Attribute: HACheckInterval
- Attribute: MlpAltitudeSupported
- Attribute: MlpDelay
- Attribute: MlpLocationEstimatesOK
- Attribute: MlpLocationReqType
- Attribute: MlpPassword
- Attribute: MlpPushAddr
- Attribute: MlpReactivateOnStartup
- Attribute: MlpRequestor
- Attribute: MlpServiceId
- Attribute: MlpSrsName
- Attribute: MlpUrl
- Attribute: MlpUser
- Attribute: MlpVersionSupported
- Attribute: OverloadPercentage
- Attribute: RequestBuffering
- Attribute: SevereOverloadPercentage
- Attribute: XMLDocTypeTagUsage
- Operation: explicitlyActivatePlugin
- Operation: registerInResourceManager
- Operation: unregisterFromResourceManager
- Operation: unregisterFromResourceManager

**Attribute: BufferFlushInterval**

Scope: Server

Unit: milliseconds
Format: int
Sets the time between each buffer flush (one unit) during recovery.
Only used if request buffering is used, see Attribute: RequestBuffering.

Attribute: BurstInterval
Scope: Server
Unit: milliseconds
Format: int
Sets the time between each call (the subdivisions of the unit) in each buffer flush during recovery
Only used if request buffering is used, see Attribute: RequestBuffering.

Attribute: BurstSize
Scope: Server
Unit: n/a
Format: int
Specifies the number of requests sent in each flush when emptying the request buffer.
Only used if request buffering is used, see Attribute: RequestBuffering.

Attribute: CharacterEncoding
Scope: Server
Unit: n/a
Format: string
Indicates the type of Unicode character encoding accepted by the MLP node. The values are not case sensitive. A typical value is UTF-8.

Attribute: HACheckInterval
Scope: Server
Unit: milliseconds
Format: int
 Specifies the time interval between MLP server heartbeat checks. Heartbeats are only sent if the connection with the MLP server is lost.

**Attribute: MlpAltitudeSupported**

Scope: Server
Unit: n/a
Format: Boolean
Specifies if the MLP server supports altitude requests. When set to true, the `<alt_acc>` tag will be included in requests towards the MLP server.

Only applicable when the plug-in operates in MLP 3.2 mode, see **Attribute: MlpVersionSupported**.

**Attribute: MlpDelay**

Scope: Server
Unit: n/a
Format: int
Specifies the default MLP response request type. If set to NOT_USED, the `<eqop>` tag will not be used in requests towards the MLP server. The following are supported:

- 0 for NO_DELAY
- 1 for LOW_DELAY
- 2 for DELAY_TOL
- 3 for NOT_USED

**Attribute: MlpLocationEstimatesOK**

Scope: Server
Unit: n/a
Format: Boolean
Specifies if the MLP server is allowed to estimate locations. Use true if estimates are allowed, otherwise false.

Defines the value of the attribute `loc_estimates` in MLP.
**Attribute: MlpLocationReqType**

Scope: Server  
Unit: n/a  
Format: int  
Specifies which type of location request to use towards the MLP server.  
Valid values are:  
- 0 for EME_LIR (Emergency location request)  
- 1 for SLIR (Standard location request)  
Defines the DTD to be used for constructing the request towards the MLP server.

**Attribute: MlpPassword**

Scope: Server  
Unit: n/a  
Format: string  
Specifies the password used when Network Gatekeeper connects to the MLP server. The password is provided by the MLP server administrator.

**Attribute: MlpPushAddr**

Scope: Server  
Unit: n/a  
Format: URL  
Specifies the callback URL to which the MLP server delivers location reports. This is the URL at which Network Gatekeeper listens for location reports. The format for the URL is:  
http://<ipaddressOfNTMachine>:<portOfWLS>/mlp/mlp_client  
For example, http://172.16.0.0:8001/mlp/mlp_client

**Attribute: MlpReactivateOnStartup**

Scope: Server  
Unit: n/a
Format: Boolean

Specifies if triggers registered by an application should be re-registered in the MLP server when Network Gatekeeper starts up.

**Attribute: MlpRequestor**

Scope: Server

Unit: n/a

Format: String

Specifies the WebLogic Network Gatekeeper requestor ID. If set to an empty string, the `<requestorid>` tag will not be used in the MLP request. The requestor ID is provided by the MLP server administrator.

**Attribute: MlpServiceId**

Scope: Server

Unit: n/a

Format: String

Specifies the WebLogic Network Gatekeeper service ID. If set to an empty string, the `<serviceid>` tag will not be used in the MLP request. The service ID is provided by the MLP server administrator.

**Attribute: MlpSrsName**

Scope: Server

Unit: n/a

Format: String

Specifies requested MLP srsName attribute.

Normally this is `www.epsg.org#4326`

**Attribute: MlpUrl**

Scope: Server

Unit: n/a

Format: URL
Specifies the MLP server’s URL

**Attribute: MlpUser**
Scope: Server
Unit: n/a
Format: String
Specifies the Network Gatekeeper user ID used when connecting to the MLP server. The user ID is provided by the MLP administrator.

**Attribute: MlpVersionSupported**
Scope: Server
Unit: n/a
Format: String [3.0.0 | 3.2.0]
Specifies which version of MLP to use.
Allowed values are:
- 3.0.0
- 3.2.0

**Attribute: OverloadPercentage**
Scope: Server
Unit: n/a
Format: int [0...100]
Specifies the load percentage at which the plug-in will raise an overload alarm. Must be less than **Attribute: SevereOverloadPercentage**.

**Attribute: RequestBuffering**
Scope: Server
Unit: n/a
Format: Boolean [TRUE|FALSE]
Specifies if request buffering should be used or not. See Attribute: RequestBuffering for information about the buffering scheme.

- If used (TRUE), the requests will be buffered if the MLP server does not respond to a request.
- If not used (i.e., FALSE), requests will be rejected if the MLP server does not respond.

**Attribute: SevereOverloadPercentage**

Scope: Server  
Unit: n/a  
Format: int [0...100]  
Specifies the load percentage at which the plug-in will raise a severely overloaded alarm.  
Must be larger than Attribute: OverloadPercentage.

**Attribute: XMLDocTypeTagUsage**

Scope: Server  
Unit: n/a  
Format: Boolean [true|false]  
Specifies if the XML tag `<!DOCTYPE>` should be included in requests towards the MLP node.  
Valid values are:

- true - include the tag  
- false - do not include the tag

**Operation: explicitlyActivatePlugin**

Scope: Server  
Activates or deactivates the plug-in. This operation is useful in situations where the heartbeat has been turned off and the plug-in is in a deactivated state.

Signature:

```java
explicitlyActivatePlugin(isActive:boolean)
```
Operation: `registerInResourceManager`
Scope: Server
Explicitly register the plug-in in the plug-in manager. This operation must be performed after a MLP server failure if Attribute: RequestBuffering is false.
Signature:
`registerInResourceManager()`

Table 21-2 `registerInResourceManager`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isActive</td>
<td>Use:</td>
</tr>
<tr>
<td></td>
<td>• true if the plug-in should be put in state activated.</td>
</tr>
<tr>
<td></td>
<td>• false if the plug-in should be put in state deactivated.</td>
</tr>
</tbody>
</table>

Operation: `unregisterFromResourceManager`
Scope: Server
Explicitly un-register the plug-in from the plug-in manager. When unregistered, the plug-in will not be visible to the plug-in manager and no requests will be passed to it.
Signature:
`unregisterFromResourceManager()`
OSA User Location plug-in for backwards compatible
Terminal Location traffic path

Managed object: Plugin_OSA_UL

Most of the configuration for the OSA User Location for Terminal Location is done in the OSA
Access module, but with configuration details for the OSA User Location SCS. See Managing
and Configuring OSA/Parlay Gateway Connections.

Configuration workflow

Below is an outline for configuring the plug-in:

1. Gather information about the OSA Gateway and configure the plug-in accordingly. The
   following information needs to be obtained from the OSA Gateway administrator and
   configured in the OSA Access service:

   - OSA/Parlay SCS type to be used in the look up (service discovery) phase when
     requesting the service (OSA/Parlay SCS) from the OSA/Parlay Gateway. Typically this
     is P_USER_LOCATION.

   - OSA/Parlay service properties to be used in the look up (service discovery) phase when
     requesting a service (OSA/Parlay SCS) from the OSA/Parlay Gateway. This depends
     on the OSA Gateway implementation.

   - Authentication type used by the OSA/Parlay Framework.

   - Encryption method used for the connection with the OSA Gateway.

   - Signing algorithm used when signing the service level agreement with the OSA/Parlay
     Framework.

---

Table 21-3 unregisterFromResourceManager

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unregisterFromResourceManager</td>
<td></td>
</tr>
</tbody>
</table>

---
2. Setup the OSA Client and the OSA Client Mappings according to Creating an OSA client and Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS in Chapter 27, “Managing and Configuring OSA/Parlay Gateway Connections.”

3. Configure the behavior of the plug-in by defining the following attributes in the OSA User Location plug-in:

   Attribute: MaxNrOfAddressesPerRequest

**Reference: attributes and operations for Plugin_OSA_UL**

Below is a list of attributes and operations for configuration and maintenance:

- Attribute: MaxNrOfAddressesPerRequest
- Attribute: NumberOfActiveSessions (r)

**Attribute: MaxNrOfAddressesPerRequest**

Scope: Server

Unit: milliseconds

Format: int

 Specifies the maximum number of address to be positioned in one single request.

**Attribute: NumberOfActiveSessions (r)**

Read-only.

Displays the number of active sessions towards the User Location SCS.
Managing and Configuring Call Handling Traffic Paths

The following section describes configuration and maintenance attributes and operations for the Parlay X 2.1 Call Handling traffic path. It also provides a workflow for the configuration:

- **OSA MultiParty Call Control and OSA Generic User Interaction plug-in for Call Handling**
  - Configuration workflow
  - Reference: attributes and operations for Plugin_call_handling_parlay_mpcc

**Call Handling Traffic Path**

**OSA MultiParty Call Control and OSA Generic User Interaction plug-in for Call Handling**

Managed object: Plugin_call_handling_parlay_mpcc

For a configuration workflow, see Configuration workflow.

Most of the configuration for the OSA MultiParty Call Control and OSA Generic User Interaction plug-in for Call Handling is done in the OSA Access module, but with configuration details for the OSA MultiParty Call Control and OSA Generic User Interaction SCS. See Managing and Configuring OSA/Parlay Gateway Connections.

**Configuration workflow**

Below is an outline for configuring the plug-in using the Network Gatekeeper Management Console:
1. Gather information about the OSA Gateway and configure the MultiParty Call Control part of the plug-in accordingly. The following information needs to be obtained from the OSA Gateway administrator and configured in the OSA Access service:

   - OSA/Parlay SCS type to be used in the look up (service discovery) phase when requesting the MultiParty Call Control service (OSA/Parlay SCS) from the OSA/Parlay Gateway. Typically this is P_MULTI_PARTY_CALL_CONTROL.

   - OSA/Parlay service properties to be used in the look up (service discovery) phase when requesting a service (OSA/Parlay SCS) from the OSA/Parlay Gateway. This depends on the OSA Gateway implementation.

   - Authentication type used by the OSA/Parlay Framework.

   - Encryption method used for the connection with the OSA Gateway.

   - Signing algorithm used when signing the service level agreement with the OSA/Parlay Framework.

2. Setup the OSA Client and the OSA Client Mappings according to Creating an OSA client and Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS in Chapter 27, “Managing and Configuring OSA/Parlay Gateway Connections.” for the MultiParty Call Control part of the plug-in.

3. Gather information about the OSA Gateway and configure the Generic User Interaction part of the plug-in accordingly. The following information needs to be obtained from the OSA Gateway administrator and configured in the OSA Access service:

   - OSA/Parlay SCS type to be used in the look up (service discovery) phase when requesting the Generic User interaction service (OSA/Parlay SCS) from the OSA/Parlay Gateway. Typically this is P_USER_INTERACTION.

   - OSA/Parlay service properties to be used in the look up (service discovery) phase when requesting a service (OSA/Parlay SCS) from the OSA/Parlay Gateway. This depends on the OSA Gateway implementation.

   - Authentication type used by the OSA/Parlay Framework.

   - Encryption method used for the connection with the OSA Gateway.

   - Signing algorithm used when signing the service level agreement with the OSA/Parlay Framework.

4. Setup the OSA Client and the OSA Client Mappings according to Creating an OSA client and Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS in Chapter 27, “Managing and Configuring OSA/Parlay Gateway Connections.” for the Generic User Interaction part of the plug-in.
5. Configure the behavior of the plug-in. In the managed object Plugin_call_handling_parlay_mpcc, the following configuration parameter can be set:
   - **Attribute: RepeatIndicator**
   - **Attribute: ResponseRequested**.

6. Setup the routing rules to the plug-in, see Administration of plug-in routes. Use the plug-in ID Plugin_call_handling_parlay_mpcc.

7. If desired, create and load a node SLA, see Updating and loading a global node SLA.

**Reference: attributes and operations for Plugin_call_handling_parlay_mpcc**

Below is a list of attributes and operations for configuration and maintenance:

- **Attribute: RepeatIndicator**
- **Attribute: ResponseRequested**
- **Operation: deleteRule**
- **Operation: getCallhandlingRule**
- **Operation: listRules**

**Attribute: RepeatIndicator**
Scope: Cluster
Unit: n/a
Format: int

Specifies the number of times a message should be played to the B-party.
The value corresponds to the parameter `repeatIndicator` in the `sendInfoReq` requests towards the Generic User Interaction SCS.

**Attribute: ResponseRequested**
Scope: Cluster
Unit: n/a
Format: int
Specifies if a response is required from the Generic User Interaction SCS, and what, if any, action the service should take.

The value corresponds to the parameter `responseRequested` in the `sendInfoReq` requests towards the Generic User Interaction SCS.

Use:

- 1 for `P_UI_RESPONSE_REQUIRED`
- 2 for `P_UI_LAST_ANNOUNCEMENT_IN_A_ROW`
- 4 for `P_UI_FINAL_REQUEST`

**Operation: deleteRule**

Scope: Cluster

Removes a rule.

Signature:

```
deleteRule(destinationURIAddress: String)
```

**Table 22-1 deleteRule**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destinationURIAddress</td>
<td>The URI that was the destination address when the rule was created. For example <code>tel:123</code></td>
</tr>
</tbody>
</table>

**Operation: getCallhandlingRule**

Scope: Cluster

Retrieves a specific rule.

Signature:

```
getRule(destinationURIAddress: String)
```
Table 22-2  getCallhandlingRule

| deleteRule |
|---|---|
| Parameter | Description |
| destinationURIAddress | The URI that was the destination address when the rule was created. For example tel:123 |

**Operation: listRules**

Scope: Cluster

Lists all the destination URIs for all the existing rules

Signature:

listRules()

Table 22-3  listRules

<p>| listRules |
|---|---|</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
Managing and Configuring Audio Call Traffic Path

The following section describes configuration and maintenance attributes and operations for the Parlay X 2.1 Audio Call path. It also provides a workflow for the configuration:

- **OSA MultiParty Call Control and OSA Generic User Interaction plug-in for Audio Call**
  - Configuration workflow
  - Reference: attributes and operations for Plugin_audio_call_parlay_mpcc_cui

### Audio Call Traffic Path

**OSA MultiParty Call Control and OSA Generic User Interaction plug-in for Audio Call**

Managed object: Plugin_audio_call_parlay_mpcc_cui

For a configuration workflow, see [Configuration workflow](#).

Most of the configuration for the OSA MultiParty Call Control and OSA Generic User Interaction plug-in for Audio Call is done in the OSA Access module, but with configuration details for the OSA MultiParty Call Control and OSA Generic User Interaction SCS. See [Managing and Configuring OSA/Parlay Gateway Connections](#).

**Configuration workflow**

Below is an outline for configuring the plug-in using Network Gatekeeper Management Console:
1. Gather information about the OSA Gateway and configure the MultiParty Call Control part of the plug-in accordingly. The following information needs to be obtained from the OSA Gateway administrator and configured in the OSA Access service:
   - OSA/Parlay SCS type to be used in the look up (service discovery) phase when requesting the MultiParty Call Control service (OSA/Parlay SCS) from the OSA/Parlay Gateway. Typically this is \texttt{P\_MULTI\_PARTY\_CALL\_CONTROL}.
   - OSA/Parlay service properties to be used in the look up (service discovery) phase when requesting a service (OSA/Parlay SCS) from the OSA/Parlay Gateway. This depends on the OSA Gateway implementation.
   - Authentication type used by the OSA/Parlay Framework.
   - Encryption method used for the connection with the OSA Gateway.
   - Signing algorithm used when signing the service level agreement with the OSA/Parlay Framework.

2. Setup the OSA Client and the OSA Client Mappings according to Creating an OSA client and Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS in Chapter 27, “Managing and Configuring OSA/Parlay Gateway Connections.” for the MultiParty Call Control part of the plug-in.

3. Gather information about the OSA Gateway and configure the Generic User Interaction part of the plug-in accordingly. The following information needs to be obtained from the OSA Gateway administrator and configured in the OSA Access service:
   - OSA/Parlay SCS type to be used in the look up (service discovery) phase when requesting the Generic User interaction service (OSA/Parlay SCS) from the OSA/Parlay Gateway. Typically this is \texttt{P\_USER\_INTERACTION}.
   - OSA/Parlay service properties to be used in the look up (service discovery) phase when requesting a service (OSA/Parlay SCS) from the OSA/Parlay Gateway. This depends on the OSA Gateway implementation.
   - Authentication type used by the OSA/Parlay Framework.
   - Encryption method used for the connection with the OSA Gateway.
   - Signing algorithm used when signing the service level agreement with the OSA/Parlay Framework.

4. Setup the OSA Client and the OSA Client Mappings according to Creating an OSA client and Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS in Chapter 27, “Managing and Configuring OSA/Parlay Gateway Connections.” for the Generic User Interaction part of the plug-in.
5. Configure the behavior of the plug-in. In the managed object AudioCallMBean, the following configuration parameter can be set:
   - Attribute: RepeatIndicator
   - Attribute: ResponseRequested.

6. Setup the routing rules to the plug-in, see Administration of plug-in routes. Use the plug-in ID Plugin_audio_call_parlay_mpcc_cui.

7. If desired, create and load a node SLA, see Updating and loading a global node SLA.

Reference: attributes and operations for Plugin_audio_call_parlay_mpcc_cui

Below is a list of attributes and operations for configuration and maintenance:

- Attribute: RepeatIndicator
- Attribute: ResponseRequested

**Attribute: RepeatIndicator**

Scope: Cluster

Unit: n/a

Format: int

Specifies the number of times a message should be played to the B-party.

The value corresponds to the parameter repeatIndicator in the sendInfoReq requests towards the Generic User Interaction SCS.

**Attribute: ResponseRequested**

Scope: Cluster

Unit: n/a

Format: int

Specifies if a response is required from the Generic User Interaction SCS, and what, if any, action the service should take.

The value corresponds to the parameter responseRequested in the sendInfoReq requests towards the Generic User Interaction SCS.
Use:

- 1 for P_UI_RESPONSE_REQUIRED
- 2 for P_UI_LAST_ANNOUNCEMENT_IN_A_ROW
- 4 for P_UI_FINAL_REQUEST

Normally P_UI_RESPONSE_REQUIRED shall be used.
Managing and Configuring the Presence Traffic Path

The following section describes configuration and maintenance attributes and operations for the Parlay X 2.1 Presence traffic path.

- Presence Traffic Path
  - SIP integration for the Parlay X 2.1 Presence traffic path

**Presence Traffic Path**

The Presence traffic path comes with support for the following network plug-ins:

- SIP integration for the Parlay X 2.1 Presence traffic path

**SIP integration for the Parlay X 2.1 Presence traffic path**

This section contains a description of the attributes and operations available for the SIP integration for the Presence traffic path.

Network Gatekeeper uses two parts to integrate with SIP networks:

One part executes in Network Gatekeeper, and one part executes as a SIP Application in a BEA WebLogic SIP Server installation. The two parts execute in different containers and require configuration in both.

For a configuration workflow, see Configuration workflow.

The plug-in uses a set of caches:

- URI cache
Managing and Configuring the Presence Traffic Path

- Subscriptions cache
- Notifications cache

**URI cache**
In Parlay X 2.1 Presence, the sip URI of the user is not passed as an argument. Instead the application (identified by the account username, also known as the application instance group ID) that is using the interface must be mapped to a URI. The URI mapping is configured as a part of the service provider application provisioning workflow and stored in the URI cache. For requests that originate from an application, the URI is fetched from this cache before being put into the from header in the SIP requests. For application terminating requests, the to header URI passed in the SIP NOTIFY requests is used to look up the account username/application instance group ID.

**Subscriptions cache**
Every subscription, pending or not, is stored in this cache during the subscription’s lifetime. It is added when an application invokes the subscribePresence method on the application-facing interface, and removed when the subscription is terminated.

**Notifications cache**
All ongoing notifications are cached. The entries are created when an application invokes startPresenceNotification on the application-facing interface and are removed when endPresenceNotification is invoked, the end criteria are reached, or the subscription is ended.

**Configuration workflow**
Below is an outline for configuring the plug-in:
In the Network Gatekeeper Management Console.

1. Decide the behavior of the traffic path by configuring the Network Gatekeeper part of the SIP integration for the Parlay X 2.1 Presence traffic path. The following configuration parameters are available in the managed object Plugin_presence_sip:
   - Attribute: DefaultNotificationCount
   - Attribute: DefaultNotificationDuration
   - Attribute: NotificationCleanupTimerValue
– **Attribute:** NotificationCleanupTimerValue
– **Attribute:** SIPNodePassword:
– **Attribute:** SIPNodeUrl
– **Attribute:** SIPNodeUsername

2. Use **Operation:** connectToWLSS to setup the connection to the part of the plug-in that executes in WebLogic SIP server.

3. Setup the routing rules to the plug-in(s), see **Administration of plug-in routes.** The plug-in ID is Plugin_presence_sip.

4. If desired, create and load a node SLA, see **Updating and loading a global node SLA.** Move on to the provisioning of service provider accounts and application accounts.

**Provisioning workflow**

For every application a mapping between a SIP URI and application instance group ID needs to be defined using **Operation:** setApplicationInstanceGroupSIPURI.

To display the mapping, use **Operation:** getApplicationInstanceGroupSIPURI.

If an application is deleted, the data for the application needs to be removed using **Operation:** removeAppinstGrpfromCache.

**Management operations**

The following operations are related to management:

- **Operation:** clearCache
- **Operation:** connectToWLSS
- **Operation:** listNotificationsCache
- **Operation:** listSubscriptionsCache
- **Operation:** listURImappingCache

**Attributes and operations for Plugin_presence_sip**

Below is a list of attributes and operations for configuration and maintenance.

- **Attribute:** DefaultNotificationCount
Managing and Configuring the Presence Traffic Path

- **Attribute: DefaultNotificationDuration**
- **Attribute: NotificationCleanupTimerValue**
- **Attribute: SubscriptionCleanupTimerValue**
- **Attribute: SIPNodePassword:**
- **Attribute: SIPNodeUrl**
- **Attribute: SIPNodeUsername**
- **Operation: clearCache**
- **Operation: connectToWLSS**
- **Operation: getApplicationInstanceGroup**
- **Operation: getApplicationInstanceGroupSIPURI**
- **Operation: listNotificationsCache**
- **Operation: listSubscriptionsCache**
- **Operation: listURImappingCache**
- **Operation: removeAppinstGrpfromCache**
- **Operation: removeNotification**
- **Operation: removeSubscription**
- **Operation: setApplicationInstanceGroupSIPURI**

**Attribute: DefaultNotificationCount**
Scope: Cluster
Unit: n/a
Format: int
Specifies the default notification count value. This value is used if none is provided in the startNotification requests from the application.

**Attribute: DefaultNotificationDuration**
Scope: Cluster
Presence Traffic Path

Unit: seconds
Format: int
Specifies the value of the default notification duration. This value is used if none is provided in the startNotification request form the application.

Example values:

- 86400 seconds is 1 day
- 604800s is 1 week

**Attribute: NotificationCleanupTimerValue**
Scope: Cluster
Unit: seconds
Format: int
Specifies the value of the timer used for checking on and cleaning up old notifications.

Each time the timer expires, it initiates a check for old notifications. If an old notification is found during the check it will be removed internally and a statusEnd callback is made to the application.

**Attribute: SubscriptionCleanupTimerValue**
Scope: Cluster
Unit: seconds
Format: int
Specifies the value of the timer used for checking on and cleaning up old subscription.

Each time the timer expires, it initiates a check for old notifications. If an old subscription is found during the check it will be removed a callback is made to the application.

**Attribute: SIPNodePassword:**
Scope: Cluster
Unit: n/a
Format: String
Specifies the password used in the JNDI lookup of WLSS.

**Note:** You need to invoke Operation: connectToWLSS to make the change take effect.
Attribute: SIPNodeUrl
Scope: Cluster
Unit: n/a
Format: String in URI format
Specifies the WebLogic SIP Server naming service containing the host name (or IP address) and the port number identifying \texttt{wlss.jndi.uri}. This was configured when WebLogic SIP server was installed.
Example: \texttt{t3://127.0.0.1:7002}
Specifies the URL to the SIP application part of the SIP integration for Presence.
\textbf{Note:} You need to invoke \texttt{Operation: connectToWLSS} to make the change take effect.
\textbf{Note:} This is not the same as the Network Gatekeeper naming service

Attribute: SIPNodeUsername
Scope: Cluster
Unit: n/a
Format: String
Specifies the user name for the SIP application part of the SIP integration for Presence. This is a regular WebLogic SIP Server user.
\textbf{Note:} You need to invoke \texttt{Operation: connectToWLSS} to make the change take effect.

\textbf{Operation: clearCache}
Scope: Cluster
Clears one or all caches used by this plug-in.
\textbf{Note:} Use this method with care.
Signature:
\texttt{clearCache(cacheToClear: String)
Table 24-1 clearCache

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacheToClear</td>
<td>Name of the cache to clear. Valid options:</td>
</tr>
<tr>
<td></td>
<td>● NOTIFICATIONS - clears the notification cache</td>
</tr>
<tr>
<td></td>
<td>● SUBSCRIPTIONS - clears the subscriptions cache</td>
</tr>
<tr>
<td></td>
<td>● URMAPPINGS - clears the URI mappings cache</td>
</tr>
<tr>
<td></td>
<td>● ALL - clears all caches.</td>
</tr>
</tbody>
</table>

Operation: connectToWLSS
Scope: Cluster
Performs a lookup of the PresenceConsumer EJB on the WebLogic SIP server side of the plug-in. Returns false if lookup failed.
Signature:
connectToWLSS()

Table 24-2 connectToWLSS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation: getApplicationInstanceGroup
Scope: Cluster
Displays the application instance group associated with a SIP URI. The application instance group, also known as the account username, is associated with an application.
Signature:
getApplicationInstanceGroup(uri: String)
Table 24-3  getApplicationInstanceGroup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>The SIP URI.</td>
</tr>
</tbody>
</table>

**Operation: getApplicationInstanceGroupSIPURI**

Scope: Cluster

Displays the SIP URI associated with an application instance group. The application instance group, also known as the account username, is associated with an application.

Signature:

```java
getApplicationInstanceGroupSIPURI(appInstGrpId: String)
```

Table 24-4  getApplicationInstanceGroupSIPURI

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appInstGrpId</td>
<td>ID of the application instance group.</td>
</tr>
</tbody>
</table>

**Operation: listNotificationsCache**

Scope: Cluster

Displays the cache where notification information is stored. Used for troubleshooting.

**Note:** Use with caution, lists data from all entries in the notification cache.

Signature:

```java
listNotificationsCache()
```
Operation: listSubscriptionsCache
Scope: Cluster
Displays the cache where subscription information is stored. Used for troubleshooting.

Note: Use with caution, lists data from all entries in the subscriptions cache.

Signature:
listSubscriptionsCache()

Table 24-6 listSubscriptionsCache

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Operation: listURImappingCache
Scope: Cluster
Displays the cache where URI mappings information is stored. Used for troubleshooting.

Note: Use with caution, lists data from all entries in the URI mappings cache.

Signature:
listURImappingCache()
Managing and Configuring the Presence Traffic Path

Table 24-7 listURImappingCache

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Operation: removeAppInstGrpFromCache

Scope: Cluster

Removes entries that are associated with an application instance group from the URI mappings cache. If an application instance group (application) has been removed, the associated entries in the cache must be removed, too.

Signature:

removeAppInstGrpFromCache()

Table 24-8 removeAppInstGrpFromCache

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appInstGrpId</td>
<td>ID of the application instance group.</td>
</tr>
</tbody>
</table>

Operation: removeNotification

Scope: Cluster

Removes a notification. The application will not be notified that the notification has been removed.

Signature:

removeNotification(appInstGrpId: String, presentity: String)
Operation: removeSubscription
Scope: Cluster

Removes a subscription and notifications. The application will not be notified that the subscription has been removed.

Signature:

removeSubscription(appInstGrpId: String, presentity: String)

Table 24-9  removeNotification

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appInstGrpId</td>
<td>ID of the application instance group.</td>
</tr>
<tr>
<td>presentity</td>
<td>ID of the presentity.</td>
</tr>
</tbody>
</table>

Operation: setApplicationInstanceGroupSIPURI
Scope: Cluster

Associates a SIP URI with an application instance group. See URI cache.

Signature:

setApplicationInstanceGroupSIPURI(appInstGrpId: String, URI: String)
## Table 24-11 setApplicationInstanceGroupSIPURI

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appInstGrpId</td>
<td>ID of the application instance group.</td>
</tr>
<tr>
<td>URI</td>
<td>SIP URI.</td>
</tr>
<tr>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td>sip:<a href="mailto:name@somedomain.org">name@somedomain.org</a></td>
</tr>
</tbody>
</table>
Managing and Configuring WAP Push Traffic Paths

The following section describes configuration and maintenance attributes and operations for the Extended Web Services WAP Push traffic paths. It also provides a workflow for the configuration:

- **Backwards compatible WAP Push traffic paths**
  - Service capability for backwards compatible WAP Push traffic path
  - PAP plug-in for WAP Push backwards compatible traffic paths
  - OSA/Parlay Multimedia Messaging plug-in for WAP Push backwards compatible traffic paths

**Backwards compatible WAP Push traffic paths**

All WAP Push related operations are handed off to network nodes that accept and forward WAP Push messages to end user terminals.

This path uses two common software modules for all traffic on the network tier, a stateless adapter and a service capability. When a request has passed these two layers, the plug-in selection is performed and the request is handed off to the network protocol plug-in. There are configuration settings for all these layers.

**Overall configuration workflow**

Below is an outline for configuring the WAP Push traffic path using the Network Gatekeeper Management Console:
1. Setup routing to the WAP Push traffic path. See Administration of plug-in routes. Use the plug-in ID Plugin_push_message_bc.

2. Set the following attributes in the managed object SESPA_message_sender, as necessary:
   - Attribute: RequestTimeout

3. Set the following attributes in the managed object ESPA_message_sender, as necessary:
   - Attribute: NotificationCdrUsage
   - Attribute: OverloadPercentage
   - Attribute: RequestTimeout
   - Attribute: SevereOverloadPercentage

4. Define which network protocol plug-in to use and follow the configuration flow for each plug-in:
   - PAP plug-in for WAP Push backwards compatible traffic paths

Move on to the provisioning of Service provider accounts and application accounts.

**Stateless adapter for backwards compatible WAP Push traffic path**

Managed object: SESPA_message_sender

**Reference: attributes and operations for SESPA_message_sender**

Below is a list of attributes and operations for configuration and maintenance:

- Attribute: RequestTimeout

**Attribute: RequestTimeout**

Scope: Cluster
Unit: Seconds
Format: int

Sets the time out value for the requests made on the interface of the module.
Service capability for backwards compatible WAP Push traffic path

Managed object: ESPA_message_sender

Attributes and operations for ESPA_message_sender

Below is a list of attributes and operations for configuration and maintenance:

- Attribute: NotificationCdrUsage
- Attribute: OverloadPercentage
- Attribute: RequestTimeout
- Attribute: NrOfPendingRequests (r)
- Attribute: SevereOverloadPercentage

Attribute: NotificationCdrUsage

Scope: Cluster
Unit: n/a
Format: Boolean

Specifies if a CDR should be generated for status notifications, that is a notification that the message has been handed off to the terminal. If set to:

- TRUE, a CDR will be generated for each status notification
- FALSE, no CDRs will be generated for status notifications

Attribute: OverloadPercentage

Scope: Server
Unit: n/a
Format: int [0...100]

Specifies the load percentage at which the software module will raise an overloaded alarm. Must be less than Attribute: SevereOverloadPercentage.
Attribute: RequestTimeout
Scope: Cluster
Unit: Seconds
Format: int
Sets the timeout value for the requests made on the interface of the module.

Attribute: NrOfPendingRequests (r)
Scope: Cluster
Unit: n/a
Format: int
Read only.
Displays the number of pending requests that are waiting for response from the network and response from the application. These request types are:

- sendMessageReq
- sendStatusNotificationReq

Attribute: SevereOverloadPercentage
Scope: Server
Unit: n/a
Format: int [0...100]
Specifies the load percentage at which the software module will raise a severely overloaded alarm.

Must be larger than Attribute: OverloadPercentage.

PAP plug-in for WAP Push backwards compatible traffic paths
Managed object: Plugin_message_sender_PAP
For a configuration workflow, see Configuration workflow.
**Configuration workflow**

Below is an outline for configuring the plug-in:

1. Configure the behavior of the plug-in by defining the following attributes in the managed object `Plugin_message_sender_PAP`. The following attributes change the characteristics can be set:
   - Attribute: `PushProxyGatewayURL`
   - Attribute: `PPGNotifyRequestedToURL`
   - Operation: `setHeartbeatConfiguration`
   - Attribute: `OverloadPercentage`
   - Attribute: `SevereOverloadPercentage`
   - Attribute: `NotificationRespTimeoutValue`

2. If desired, create and load a node SLA, see Updating and loading a global node SLA.

**Reference: attributes and operations for Plugin_message_sender_PAP**

Below is a list of attributes and operations for configuration and maintenance:

- Attribute: `OverloadPercentage`
- Attribute: `PushProxyGatewayURL`
- Attribute: `SevereOverloadPercentage`
- Attribute: `PPGNotifyRequestedToURL`
- Attribute: `NotificationRespTimeoutValue`
- Attribute: `HeartbeatConfiguration` (r)
- Operation: `setHeartbeatConfiguration`

**Attribute: OverloadPercentage**

Scope: Server

Unit: n/a

Format: int [0...100]
Specifies the load percentage at which the software module will raise an overload alarm. Must be less than Attribute: SevereOverloadPercentage.

**Attribute: PushProxyGatewayURL**
Scope: Server
Unit: n/a
Format: String
Specifies the URL to the Push Proxy Gateway (PPG).

**Attribute: SevereOverloadPercentage**
Scope: Server
Unit: n/a
Format: int [0...100]
Specifies the load percentage at which the software module will raise a severely overloaded alarm.
Must be larger than Attribute: OverloadPercentage.

**Attribute: PPGNotifyRequestedToURL**
Scope: Server
Unit: n/a
Format: String
Specifies the URL that the Push Proxy Gateway (PPG) should use for notification of results. This is the ppg-notify-requested-to attribute of the push-message operation.

**Attribute: NotificationRespTimeoutValue**
Unit: milliseconds
Format: Int
Specifies the maximum time to allow for processing a result notification message from the Push Proxy Gateway (PPG). This includes processing time in the application the notification is sent to. If the processing time is longer, an error is reported to the PPG.
Attribute: HeartbeatConfiguration (r)

Read only.

Displays the current setting for heartbeat URL and heartbeat interval. See Operation: setHeartbeatConfiguration.

Operation: setHeartbeatConfiguration

Signature:

\[
\text{setHeartbeatConfiguration}(\text{config.heartbeatURL: String, config.heartbeatInterval: long})
\]

Sets the heartbeat configuration. This setting is plug-in instance specific.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>config.heartbeatURL</td>
<td>The URL to use for heartbeats.</td>
</tr>
<tr>
<td>config.heartbeatInterval</td>
<td>The time interval between heartbeats.</td>
</tr>
<tr>
<td></td>
<td>Unit: milliseconds.</td>
</tr>
<tr>
<td></td>
<td>To disable the heartbeat mechanism, set the heartbeat interval to 0.</td>
</tr>
</tbody>
</table>

OSA/Parlay Multimedia Messaging plug-in for WAP Push backwards compatible traffic paths

Managed object: Plugin_OSA_MM_WAP

For a configuration workflow, see Configuration workflow.

Configuration workflow

Below is an outline for configuring the plug-in:

1. Gather information about the OSA Gateway and configure the plug-in accordingly. The following information needs to be obtained from the OSA Gateway administrator and configured in the OSA Access service:
Managing and Configuring WAP Push Traffic Paths

- OSA/Parlay SCS type to be used in the look up (service discovery) phase when requesting the service (OSA/Parlay SCS) from the OSA/Parlay Gateway.
- OSA/Parlay service properties to be used in the look up (service discovery) phase when requesting a service (OSA/Parlay SCS) from the OSA/Parlay Gateway. This depends on the OSA Gateway implementation.
- Authentication type used by the OSA/Parlay Framework.
- Encryption method used for the connection with the OSA Gateway.
- Signing algorithm used when signing the service level agreement with the OSA/Parlay Framework.

2. Setup the OSA Client and the OSA Client Mappings according to Creating an OSA client and Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS.

3. Configure the behavior of the plug-in by defining the following attributes in the managed object Plugin_OSA_MM_WAP:
   - Attribute: MaxNumberOfMtRetries
   - Attribute: MoDecodingCharset
   - Attribute: NotificationRestoreInterval

Configure other settings using the following operations in the managed object Plugin_OSA_MM_WAP:
   - Operation: addScsProperty
   - Operation: disableNotificationRestoration or Operation: enableNotificationRestoration
   - Operation: setMtRetryInterval

4. Setup the routing rules to the plug-ins, see Administration of plug-in routes. The plug-in ID to use is generated, in numeric form. The ID is presented next to the value Plugin_OSA_MM_WAP, see Operation: listPluginTypes.

5. If desired, create and load a node SLA, see Updating and loading a global node SLA.

Reference: attributes and operations for Plugin_OSA_MM_WAP

Below is a list of attributes and operations for configuration and maintenance:

- Attribute: MaxNumberOfMtRetries
- Attribute: MoDecodingCharset
• Attribute: MTRetryInterval
• Attribute: NotificationRestoreInterval
• Operation: addScsProperty
• Operation: deleteScsProperty
• Operation: disableNotificationRestoration
• Operation: enableNotificationRestoration
• Operation: setMtRetryInterval
• Operation: disableNotification
• Operation: getNotification
• Operation: listNotifications
• Operation: listScsProperties

**Attribute: MaxNumberOfMtRetries**
Scope: Cluster
Unit: n/a
Format: int

Specifies the maximum number of times a mobile terminated message should be resent to the OSA/Parlay Gateway in case the message send operation fails.

**Attribute: MoDecodingCharset**
Scope: Cluster
Unit: n/a
Format: String

Specifies the default character set used by the OSA Gateway for mobile originated MMSes.
Enter the Java-supported encoding scheme, for example:
• ASCII
• Cp1252
• ISO8859_1
See [http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html](http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html)
Attribute: MTRetryInterval
Scope: Cluster
Unit: seconds
Format: int

Specifies the time between each retry of sending failed mobile terminated messages to the OSA Gateway.

Attribute: NotificationRestoreInterval
Scope: Cluster
Unit: seconds
Format: String

Specifies the time between each notification restoration process. See Operation: enableNotificationRestoration.

Operation: addScsProperty

Adds an OSA service property. OSA service properties are used in the lookup (service discovery) phase when requesting a service (OSA SCS) from the OSA Gateway. Duplicate name-value pair combinations cannot be added, whereas a single name and multiple values are allowed.

Signature:
addScsProperty(name: String, value: String)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the service property. Example service version.</td>
</tr>
<tr>
<td>value</td>
<td>The value of the property. Example 4.1</td>
</tr>
</tbody>
</table>

Operation: deleteScsProperty

Deletes a service property. Both name and value must match.
Signature:

```java
deleteScsProperty(name: String, value: String)
```

**Table 25-3  deleteScsProperty**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the service property to be deleted.</td>
</tr>
<tr>
<td>value</td>
<td>The value of the service property to be deleted.</td>
</tr>
</tbody>
</table>

**Operation: disableNotificationRestoration**

Disables automatic restoration of notifications.

Signature:

```java
disableNotificationRestoration()
```

**Table 25-4  disableNotificationRestoration**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Operation: enableNotificationRestoration**

Enables automatic restoration of notifications.

Signature:

```java
enableNotificationRestoration()
```
Operation: `setMtRetryInterval`

Signature:

```plaintext
setMtRetryInterval(retryInterval:int)
```

Table 25-6  `setMtRetryInterval`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>retryInterval</td>
<td>Specifies the time interval between re-send attempts of mobile terminated messages. Given in seconds.</td>
</tr>
</tbody>
</table>

Operation: `disableNotification`

Disables a previously made OSA notification.

Signature:

```plaintext
disableNotification(id: int)
```

Table 25-7  `disableNotification`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>ID of the notification to disable.</td>
</tr>
</tbody>
</table>

Table 25-5  `enableNotificationRestoration`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Managing and Configuring WAP Push Traffic Paths
**Operation: getNotification**

Returns information about specific OSA/Parlay notification.

Signature:

```java
getNotification(id: int)
```

---

**Operation: listNotifications**

Displays a list of all enabled OSA/Parlay notification IDs.

Signature:

```java
listNotifications()
```

---

**Operation: listScsProperties**

Displays a list of all service property pairs.

The format is `<property name 1>/<value 1>...<property name n>/<value n>`

Signature:

```java
listScsProperties()
```
### Table 25-10  listScsProperties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Managing and Configuring routes and Node SLAs

The following section describes configuration and maintenance attributes and operations for the Plug-in manager. It also provides a workflow for the configuration:

- Introduction
  - Execution and evaluation flow
  - How address ranges are specified
- Configuration workflow
- Writing Node SLAs
  - SLA file overview
  - Service provider node SLA
  - Node contract data
  - Global contract data
  - Node SLA tags

Introduction

Network protocol plug-ins are identified by their plug-in ID. Plug-ins also have a type which identifies to which application-facing interface they belong. The network protocol plug-ins register themselves with the plug-in manager when they are started. They register a set of properties:
Managing and Configuring routes and Node SLAs

- Plug-in type
- Plug-in ID (in some cases they are assigned a plug-in ID)
- Address plan, that is the kind of network to which they are connected: for example E.164 of SIP.

Refer to the configuration sections by traffic path for information on which plug-in ID the plug-in is registered under. Refer to the sections by traffic path for information on which type the plug-in is.

Some traffic paths, the backwards compatible traffic paths, have a layered plug-in structure. In these traffic paths the larger network plug-in (3.0 style) embeds two common modules (the stateless adapter layer and the service capability layer) and the network protocol plug-in (2.2 style). Refer to the traffic path descriptions for information on which traffic paths these are.

The backwards compatible traffic paths have two separate plug-in IDs and two separate plug-in types. For these traffic paths, a routing decision is taken twice:

1. When the request is handed over from the application tier, in order to route the request to the common modules.

   The plug-in ID used at this point is in alphanumeric for alphanumeric, for example Plugin_mmm_bc, and the plug-in type is formatted as a fully qualified Java class, for example com.bea.wlcp.wln.g.px21.plugin.SendMessagePlugin.

2. When the request is to be handed off from the common modules to the protocol-specific module.

   The plug-in ID used at this point is in IDs in the form of a number (positive or negative), for example -542200288, and the plug-in type is formatted in alphanumeric form, for example MMS_TYPE. This plug-in ID is assigned when the plug-in registers itself with the plug-in manager. Even if the IDs are assigned they are persisted so they will have the same ID if there is a server restart.

**Execution and evaluation flow**

When an application’s request is processed, it is routed to the network protocol plug-in that handles that specific kind of request and that specific address range and address plan. Routing to the appropriate type of plug-in is straightforward, because each plug-in implements the interface of only one type of application-facing interface. Multiple plug-ins may be used to cover different address ranges, so the plug-in manager must be configured to select among plug-ins based on those ranges. Routes are specified using regular expressions that match addresses to plug-ins.
When a request reaches the plug-in manager, the request is evaluated and the plug-in selection is done based on these evaluations:

1. The plug-in manager finds all plug-ins for the particular application-facing interface, that is the plug-in type.

2. Based on round-robin the plug-in manager will find the next matching plug-in based on:
   - The status of the plug-in (only plug-ins in status active are considered).
   - The supported address scheme. The type of address found in the first destination address of the request from the application is matched against the address types the plug-ins support.
   - The OAM configured address routes. The address found in the first destination address of the request from the application is matched against the pattern provided in the registered routes.

If no matching plug-in is found a **NoAvailablePluginException** is thrown and alarms are generated.

The execution flow for the plug-ins that are split up into common modules and protocol specific plug-ins perform are exposed to a second routing decision, based on the same procedure as described above.

When a plug-in of this backwards compatible type is finally selected it may execute in a server other than the one in which the common modules for that traffic path execute.

**How address ranges are specified**

Address ranges are specified using UNIX regular expressions, a few examples are given below:

- `^.*` specifies a route that matches all addresses.
- `^[0-5].*$` specifies a route that matches all address strings starting with 0, 1, 2, 3, 4, or 5.
- `^.*[6-9]$` specifies a route that matches all address strings ending with 6, 7, 8, or 9.
- `^46.*` specifies a route that matches all address string starting with 46.
- `^46.{8}$` specifies a route that matches all address strings starting with 46 that contain exactly 10 total digits.
- `^.*@.*\..com$` specifies a route matching all mail addresses in the com domain. Note that the dot in .com must be written “\.”.
Managing and Configuring routes and Node SLAs

In the examples:

- ^ indicates the beginning of the string.
- . matches anything except a new-line. That is, “a.b” matches any three-character string which begins with a and ends with b.
- * is a suffix which means the preceding regular expression is to be repeated as many times as possible. That is, in the expression “^46.*” the “.” is repeated until the whole string is matched.
- $ is a indicator of end of line (or end of string).

Address scheme can also be included in the expression. This is only supported for enhanced plug-ins and not for backwards compatible plug-ins. For example:

- tel:^46.* matches all phone numbers starting with 46.
- sip:.* matches any SIP address.

Configuration workflow

Configuring the plug-in manager can be divided into three parts:

- Configuration of the general behavior of the plug-in manager, see Configuring the plug-in manager.
- Administration of routes, which is tightly coupled to the configuration of the individual traffic paths, see Administration of plug-in routes.
- Administration of Global Node SLAs, see Updating and loading a global node SLA

Administration of plug-in routes

Administration of routes are basically:

- Operation: addRoute
- Operation: removeRoute

To change a route, remove an existing one and create a new.

Existing routes are listed using Operation: listRoutes.

To get information about a specific plug-in, use Operation: getPluginInfo.
All registered plug-in IDs are listed using Operation: listPluginIds, and all registered plug-in types are listed using Operation: listPluginTypes.

**Updating and loading a global node SLA**

A global node SLA defines allowed usage patterns for requests towards a given network node. Each instance of a plug-in on which limitations are to be placed is assigned a plug-in node ID. This ID is used when creating the global node SLA. The following operations are related to the global node SLA:

- **Operation: setPluginNodeId**
- **Operation: setNodeSlaString**
- **Operation: setNodeSlaUrl**

**Configuring the plug-in manager**

Below is an outline for configuring the plug-in manager:

1. Decide whether to use policy based routing or not. Policy based routing is necessary in order to enforce node SLAs.
   - **Attribute: PolicyBasedRouting**
   - **Attribute: ReqCounterIncrementUsage**

**Reference: attributes and operations for Plugin_manager**

Below is a list of attributes and operations for configuration and maintenance.

**Attribute: PolicyBasedRouting**

Scope: Cluster

Unit: n/a

Format: Boolean

Specifies if policy based routing should be used or not.

**Note:** Policy based routing must be enabled in order to enforce node SLAs and to be able to use custom policies for selection of routes.

Use:
Managing and Configuring routes and Node SLAs

- **true** to enable policy based routing
- **false** to disable policy based routing

**Attribute: ReqCounterIncrementUsage**
Scope: Cluster
Unit: n/a
Format: Boolean

Specifies how request counters are incremented. The counters are used for counting requests when enforcing node SLAs and when enforcing custom policies related to the plug-in manager.

Use:
- **true** to increment request counters with a number equal to the number of target addresses in the request.
- **false** to increment the request counter with one per request, regardless of the number of target addresses in the request

**Attribute: ResourceMgrIOR (r)**
Read only.
Scope: Cluster
Displays the IOR of the plug-in manager.

**Operation: addRoute**
Scope: Cluster
Adds a new plug-in route. A route is identified by the plug-in ID and the pattern.

Signature:
```
addRoute(id:String, addressExpression: String)
```
Table 26-1  addRoute

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>ID of the plug-in that is the target of the route. A list of plug-in IDs is</td>
</tr>
<tr>
<td></td>
<td>displayed in the Operation: listPluginIds attribute.</td>
</tr>
<tr>
<td>addressExpression</td>
<td>The pattern to be used as a matching criteria. See How address ranges are</td>
</tr>
<tr>
<td></td>
<td>specified.</td>
</tr>
</tbody>
</table>

**Operation: deleteNodeSla**

Scope: Cluster

Deletes the global node SLA (total traffic SLA).

Signature:

deleateNodeSla()

Table 26-2  deleteNodeSla

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: getPluginInfo**

Scope: Cluster

Displays information about a plug-in. The information includes state, which address plan the plug-in supports, routes for the plug-in and other properties. If a node ID has been assigned to the plug-in, this ID is also shown.

Signature:

getPluginInfo(id: String)
Managing and Configuring routes and Node SLAs

Table 26-3 getPluginInfo

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>ID of the plug-in.</td>
</tr>
</tbody>
</table>

Operation: listNodeSLA

Scope: Cluster

Displays the content of the global node SLA (total traffic SLA).

Signature:

listNodeSLA()

Table 26-4 listNodeSLA

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Operation: listPluginIds

Scope: Cluster

Displays a list of plug-in IDs. A plug-in ID uniquely identifies a plug-in. Identical plug-ins executing on different servers all share the same plug-in ID.

Signature:

listPluginIds()
Operation: listPluginTypes
Scope: Cluster

Displays a list of plug-in types. The plug-in type identifies which application-facing interface the plug-in implements. In the case of backwards compatible traffic path it is a name with the suffix _TYPE, for example MMS_TYPE, and in the case of a common part it identifies the name of the common part, for example com.bea.wlcp.wlng.px21.plugin.SendSmsPlugin.

Signature:
listPluginTypes()

---

Table 26-6  listPluginTypes

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listPluginTypes</td>
<td></td>
</tr>
</tbody>
</table>

Operation: listRoutes
Scope: Cluster

Displays a list of all registered routes.

Signature:
listRoutes()
Managing and Configuring routes and Node SLAs

Table 26-7 listRoutes

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: registerResource (deprecated)**

Deprecated. Do not use.

**Operation: registerResourceFromURL (deprecated)**

Deprecated. Do not use.

**Operation: removeRoute**

Scope: Cluster

Removes a route. The route is identified by the ID of the plug-in and the matching pattern.

Signature:

```
removeRoute(id: String, addressExpression: String)
```

Table 26-8 removeRoute

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>ID of the plug-in.</td>
</tr>
<tr>
<td>addressExpression</td>
<td>The pattern used as a matching criteria. See How address ranges are specified.</td>
</tr>
</tbody>
</table>

**Operation: saveResourceMgrIOR (deprecated)**

Deprecated. Do not use.
**Operation: setNodeSlaString**
Scope: Cluster
Loads a new global node SLA into memory and to persistent storage.
Signature:

```
setNodeSlaString(SLA: String)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLA</td>
<td>Contents of the global node SLA as a string of characters.</td>
</tr>
</tbody>
</table>

**Operation: setNodeSlaUrl**
Scope: Cluster
Loads a new global node SLA into memory and to persistent storage.
Signature:

```
setNodeSlaUrl(url: String)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>URL to the file that contains the global node SLA.</td>
</tr>
</tbody>
</table>

**Operation: setPluginNodeId**
Scope: Cluster
Assigns a node ID to a plug-in.
Signature:

```
setPluginNodeId(id: String, nodeId: String)
```
Managing and Configuring routes and Node SLAs

Operation: unregisterResource (deprecated)
Deprecated. Do not use.

Operation: unregisterResourceFromURL (deprecated)
Deprecated. Do not use.

Writing Node SLAs

SLA file overview
The node SLA files are written in XML. The XML SLA file looks slightly different depending on the level on which they are used, total traffic or service provider traffic. The schema for the SLAs are found in the following directory:

<Domain Home>/policy/sla_schema

Service provider node SLA
The service provider node SLA file consists of an <sla> tag containing one or more <nodeContract> tags as shown in Listing 26-1. The serviceProviderGroupID attribute specifies the service provider group for which the SLA file is valid. The structure and contents of the <nodeContract> tag is further described in Global contract data.

Listing 26-1  Service provider node SLA file

<Sla serviceProviderGroupID="spGroup1"
<nodeContract>

<!--Contract data for network node 1-->  

</nodeContract>

<nodeContract>

<!--Contract data for network node 2-->  

</nodeContract>

<nodeContract>

<!--Contract data for network node n-->  

</nodeContract>

</Sla>
**Node contract data**

The `<nodeContract>` tags contains a set of tags that define under which conditions a network node can be accessed by a service provider. See Listing 26-2. The contents of each tag are further described below the listing.

**Listing 26-2  <nodeContract> tag contents**

```xml
<nodeContract>
  <startDate>2005-01-01</startDate>
  <endDate>2010-06-01</endDate>
  <nodeID>A</nodeID>
  <nodeRestrictions>
    <nodeRestriction>
      <reqLimit>10</reqLimit>
      <timePeriod>1000</timePeriod>
    </nodeRestriction>
  </nodeRestrictions>
</nodeContract>
```

**Node SLA file**

The node SLA consists of an `<sla>` tag containing one or more `<globalContract>` tags as shown in Listing 26-3. In this case, the `serviceProviderGroupID` attribute is left empty. The structure and contents of the `<globalContract>` tag are further described in Global contract data.

**Listing 26-3  Node SLA file**

```xml
<Sla serviceProviderGroupID=""/>
Global contract data

The `<globalContract>` tags contain a set of tags that define under which conditions a network node can be accessed. The contents of each tag are further described below in Listing 26-4.
Managing and Configuring routes and Node SLAs

Listing 26-4  <globalContract> tag contents

<globalContract>
  <startDate>2005-01-01</startDate>
  <endDate>2010-06-01</endDate>
  <nodeID>A</nodeID>
  <globalRestrictions>
    <globalRestriction>
      <reqLimit>1000</reqLimit>
      <timePeriod>10000</timePeriod>
    </globalRestriction>
  </globalRestrictions>
</globalContract>

Node SLA tags

<startDate>
This tag specifies the date the service provider or WebLogic Network Gatekeeper can start accessing the network node. Use format YY-MM-DD.

<endDate>
This tag specifies the last date (expiry date) the service provider or WebLogic Network Gatekeeper can access the network node. Use format YY-MM-DD.

<nodeID>
This tag specifies the network node's node ID as registered in the Plug-in manager. Registered nodes can be listed using Operation: getPluginInfo.

<nodeRestrictions>
This tag contains one node restriction tag including sub-tags. See below.
<nodeRestriction>
This tag contains sub-tags that specify how much traffic a specific service provider is allowed to send towards the network node and under which conditions. See below.

<globalRestrictions>
This tag contains one node restriction tag including sub-tags. See below.

<globalRestriction>
This tag contains sub-tags that specify how much traffic the WebLogic Network Gatekeeper is allowed to send towards the network node and under which conditions. See below.

<reqLimit>
This tag specifies a number of requests. It is used to restrict the number of service requests allowed during a specified time period.

<timePeriod>
This tag specifies the time period (in milliseconds) during which the request limit is valid.
Managing and Configuring routes and Node SLAs
Managing and Configuring OSA/Parlay Gateway Connections

The following sections describe how to add connections to OSA/Parlay Gateways:

- Understanding OSA/Parlay Gateway and account mappings
  - Connection model
  - Information and certificate exchange with OSA/Parlay administrator
- Overall workflow when connecting to an OSA Gateway
  - Adding an OSA Gateway
  - Adding an OSA Gateway Connection
  - Creating an OSA client
  - Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS
  - Reference: attributes and operations for Plugin_OSA_Access

Understanding OSA/Parlay Gateway and account mappings

Connection model

Network Gatekeeper traffic paths use an internal service, the OSA Access service, to manage all connections with OSA/Parlay Gateways. A plug-in that connects to an OSA/Parlay SCS asks the OSA Access service for a connection, and the service handles all of the details of Authentication,
Managing and Configuring OSA/Parlay Gateway Connections

Service Discovery, and Load Management towards the OSA/Parlay Framework before returning the handle for the SCS to the plug-in.

The following concepts are used when connecting a plug-in to an OSA/Parlay Gateway:

- An **OSA Gateway**, identified by a *gatewayId*, which represents the actual OSA/Parlay Gateway. Each OSA Gateway that is used is registered in Network Gatekeeper. Any certificate to be used when authenticating with the OSA/Parlay framework is associated with the gatewayId.

- Each OSA/Parlay Gateway has one or more **OSA Gateway Connections**, identified by a *connectionID*. Multiple connections are used if the actual OSA/Parlay Gateway contains more than one Framework. The link between the OSA Gateway and the OSA Gateway connection is the gatewayID and gwID.

- An **OSA client** represents the account in the OSA/Parlay Gateway. An OSA client has the following attributes:
  - OSA client application ID, constituted of the Enterprise Operator ID and the Application ID as provisioned in the OSA/Parlay Gateway,
  - Depending on the authentication method used, a private key (with associated password and keystore password) and public certificate to be used when authenticating.

- An **OSA client mapping** maps an OSA client with OSA/Parlay SCSs. There must be (at least) one OSA client mapping per OSA SCS being used. If the traffic path uses *n* OSA SCSs, *n* Client Mappings must be defined. Three different models are possible for the OSA Client Mapping:

  a. The OSA client mapping can use wild cards for both the service provider and the application level, so all applications from all service providers are mapped to a single OSA Client. In this case, transactions in the OSA/Gateway are traceable only to Network Gatekeeper since Network Gatekeeper, from the OSA/Parlay Gateway’s viewpoint, acts as one single application.

  b. The OSA client mapping can use a wildcard for the application level, but specify the service provider, so multiple Network Gatekeeper applications that originate from a common service provider are mapped to a single OSA client. In this case, the transactions in the OSA/Gateway are traceable only to the service provider since Network Gatekeeper, from the OSA/Parlay Gateway’s viewpoint, acts as one application per service provider.

  c. The OSA client mapping may be set up per application level, so there is a one to one mapping between a Network Gatekeeper service provider and application account combination and the equivalent OSA Client. This means that every transaction originating
from a specific application results in a transaction in the OSA Gateway that is traceable to that specific application since Network Gatekeeper, from the OSA/Parlay Gateway’s viewpoint, acts as one application per service provider and application combination.

**Note:** Combinations of the above are not allowed. The Network Gatekeeper administrator must choose one of these connection modes, and use the same mode for all Network Gatekeeper applications. In the first case, the connection is a system-wide configuration, in the other two cases, the connection is setup as a part of the provisioning chain for Network Gatekeeper service providers and their applications.

Defining the OSA client mapping is a part of the provisioning chain in when setting up service provider and application accounts if the OSA client mapping is of type b. or type b.

Each OSA Client mapping has a state. The state can be:

- **Active**, which means that the connection between Network Gatekeeper and a specific SCS in the OSA Gateway is active and functional.

- **Inactive**, which means that there is no active connection. This may be because the OSA Client mapping is not configured to be initialized at startup and no requests have yet been passed to it. It may also indicate that there is a problem with the connection.

**Information and certificate exchange with OSA/Parlay administrator**

The OSA/Parlay Gateway administrator must provide the following information with regard to the OSA/Parlay Gateway account and OSA/Parlay Framework:

- The **entOpId** (Enterprise Operator ID) - Depending on how the OSA/Parlay operator administers applications (OSA/Parlay clients) the entOpId can be valid for:
  - All applications registered in WebLogic Network Gatekeeper
  - All applications connected to a service provider account
  - A single application account

- The **appId** (Application ID) to be used for the application account (clientAppId=entOpId + appId)

- The OSA/Parlay **service types** for the OSA/Parlay SCSs to which the application is to be mapped

- The encryption method used

- The signing algorithm used
Connection information for the OSA/Parlay Framework, either:
- name service reference file to the OSA/Parlay Gateway Framework’s Parlay IpInitial object.
- The name of the initial object in the name service and the file containing the IOR to the IpInitial object.

If the authentication method towards the OSA/Parlay Framework requires a certificate, the Network Gatekeeper administrator must generate one, and distribute it to the OSA/Parlay Gateway administrator. The associated key must be stored in the Network Gatekeeper keystore, this is done when the OSA client is created, see Creating an OSA client.

For non-production environments, the WebLogic Server CertGen utility can be used to create certificates and keys.

**Overall workflow when connecting to an OSA Gateway**

Follow the steps below for information on how to connect an application account to an OSA/Parlay Gateway:

1. Create a logical representation of the OSA/Parlay Gateways to connect to, see Adding an OSA Gateway.
2. For each Framework in the OSA/Parlay Gateway, create a logical representation of the Framework, see Adding an OSA Gateway Connection.
3. Define how Network Gatekeeper connects to the OSA/Parlay Gateway.
   a. If Network Gatekeeper connects to the OSA/Parlay Gateway as one single user, register this user, see Creating an OSA client.
   b. If Network Gatekeeper connects to the OSA/Parlay Gateway as several users, the registration of users is a part of the provisioning flow for service providers and applications.
4. The registration of which SCSs to use in the OSA/Parlay Gateway is done either as a part of the configuration flow for the traffic paths, or as a part of the provisioning flow for service providers and application. The procedure is described in Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS, and the data to be used is described in the configuration section for each traffic path.
Adding an OSA Gateway

An OSA Gateway is the entity representing an OSA/Parlay Gateway. One or more OSA Gateway Connections can be associated with the OSA Gateway.

1. Get the certificate for the OSA/Parlay Gateway from the administrator of the OSA/Parlay Gateway and store it on the local file system of the Network Gatekeeper’s administration server.

2. Starting in the configuration and operations page for Plugin_OSA_access, select addGw from the Select An Operation drop-down list.
   The parameters for the operation are displayed.

3. Enter the information specified in Operation: addGw

4. Click Invoke.
   The OSA Gateway is created. An ID for the OSA Gateway is returned.

Adding an OSA Gateway Connection

An OSA Gateway connection is the entity representing an individual Framework in an OSA/Parlay Gateway.

1. Get either information about how to obtain a reference to the OSA/Parlay Framework from the administrator of the OSA/Parlay Gateway. Two options are possible:
   a. The name service reference file. Store the file on the local file system of the Network Gatekeeper’s administration server.
   b. The name of the initial object in the name service and the file containing the IOR to the Parlay initial object. Store the file on the local file system of the Network Gatekeeper’s administration server.

2. Starting in the configuration and operations page for Plugin_OSA_Access, select addConnection from the Select An Operation drop-down list.
   The parameters for the operation are displayed.

3. Enter the information specified in Operation: addConnection

4. Click Invoke.
   The OSA Gateway Connection is created. An ID for the OSA Gateway is returned.
Creating an OSA client

The OSA client is the entity being used when creating the OSA client mapping.

1. Create, or get from a Certificate Authority, the private key and certificate for the client and store them on the local file system of the Network Gatekeeper’s administration server.

2. Starting in the configuration and operations page for OSA_access, select addClient from the Select An Operation drop-down list.
   The parameters for the operation are displayed.

3. Enter the information specified in Operation: addClient

4. Click Invoke.
   The OSA client is created.

Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS

The mapping may be applied on service provider account, application account, or Network Gatekeeper level.

Note: One mapping must be created for each OSA/Parlay SCS (network service) the Network Gatekeeper application is using in the OSA/Parlay gateway.

1. Starting in the configuration and operations page for Plugin_OSA_Access, select addMapping from the Select An Operation drop-down list.
   The parameters for the operation are displayed.

2. Enter the information specified in Operation: addMapping

3. Click Invoke.
   The OSA Client Mapping is created.

Reference: attributes and operations for Plugin_OSA_Access

Below is a list of attributes and operations for configuration and maintenance for Plugin_OSA_Access:
Reference: attributes and operations for Plugin_OSA_Access

- **Attribute: EricssonAuthentication**
- **Operation: addClient**
- **Operation: addConnection**
- **Operation: addGw**
- **Operation: addMapping**
- **Operation: listActiveMappings**
- **Operation: listActiveMappingsForGw**
- **Operation: listGw**
- **Operation: listMappings**
- **Operation: removeClient**
- **Operation: removeConnection**
- **Operation: removeGw**
- **Operation: removeMapping**
- **Operation: setKeyStorePassword**
- **Operation: viewActiveMappingState**

**Attribute: EricssonAuthentication**

Boolean

Set to True if connecting to an Ericsson OSA Gateway, otherwise False.

Displays a list of active OSA Client mappings.

**Operation: addClient**

Scope: Server

Adds an OSA Client.

Signature:

```
addClient(osaClientAppId: String, clientKeyFile: String, clientCertFile: String, clientKeyPwd: String, keystorePwd: String)
```
Managing and Configuring OSA/Parlay Gateway Connections

Table 27-1  addClient

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>osaClientAppId</td>
<td>The Enterprise Operator ID and Application ID registered for the OSA Client in the OSA/Parlay Gateway. This value must be unique. The format is:</td>
</tr>
<tr>
<td></td>
<td>&lt;Enterprise Operator&gt;&lt;Application ID&gt;</td>
</tr>
<tr>
<td></td>
<td>Example: myEntopId\myAppId</td>
</tr>
</tbody>
</table>

clientKeyFile The directory path (including file name) to the private key for the OSA Client.

Note: This path is on the file system of the Network Gatekeeper administration server.

clientCertFile The directory path (including file name) to the certificate for the OSA Client. The certificate is provided in order to verify the private key is correct.

Note: This path is on the file system of the Network Gatekeeper administration server.

clientKeyPwd The password for the private key.

keystorePwd The keystore’s password as defined when configuring the WebLogic Network Gatekeeper, see Operation: setKeyStorePassword.

**Operation: addConnection**

Scope: Server

Adds a connection to a Framework in the OSA Gateway.

Signature:
addConnection(gwId: int, nsRef: String, nsName: String, initialRef: String, priority: int)

**Table 27-2 addConnection**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gwId</td>
<td>The ID of the OSA/Parlay Gateway, as returned when the OSA Gateway was created. See Operation: addGw. Also see Operation: listGw.</td>
</tr>
<tr>
<td>nsRef</td>
<td>The directory path (including file name) for the file containing the name service IOR. Leave blank if initialRef is specified.</td>
</tr>
<tr>
<td>nsName</td>
<td>The name of the initial object in the name service. Example: parlay_initial. Use path syntax to specify recursive naming contexts. Example: /parlay/fw/parlay_initial Leave blank if initialRef is specified.</td>
</tr>
<tr>
<td>initialRef</td>
<td>The directory path, including file name, for the file containing the IOR to the Parlay initial object. Leave blank if nsRef and nsName is specified.</td>
</tr>
<tr>
<td>priority</td>
<td>Priority of this connection. Should be unique across all connections. The lower the number, the higher the priority.</td>
</tr>
</tbody>
</table>

**Operation: addGw**

**Scope:** Server

Adds an OSA Gateway to be used by the OSA/Parlay type plug-ins. More than one OSA Gateway can be added.

**Signature:**

`addGw(name: String, osaFwCert: String, reAuthWaitTime: int, keystorePwd: String)`
Managing and Configuring OSA/Parlay Gateway Connections

**Table 27-3 addGw**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gateway.name</td>
<td>Descriptive name of the OSA Gateway.</td>
</tr>
<tr>
<td>osaFwCert</td>
<td>The certificate to use when connecting to the OSA Gateway’s Framework. The certificate is supplied by the OSA Gateway administrator.</td>
</tr>
<tr>
<td>reAuthWaitTime</td>
<td>The time to wait before reattempting to authenticate and obtain OSA Service Managers if all connections to the OSA Gateway are lost. Given in seconds</td>
</tr>
<tr>
<td>keystorePwd</td>
<td>The password for the Network Gatekeeper keystore.</td>
</tr>
</tbody>
</table>

**Return value**

The ID for the OSA Gateway.

Used as an ID when creating an OSA Gateway Connection, see **Operation: addConnection**.

Used as an ID when creating an OSA Client Mapping, see **Operation: addMapping**.

**Operation: addMapping**

**Scope: Server**

Adds an OSA client mapping.

**Signature:**

addMapping(serviceProviderID: String, applicationID: String, serviceType: String, osaClientAppId: String, properties: String, authType: String, encryptionMethod: String, signingAlgorithm: String, gatewayId: int, initConnection: boolean)
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| serviceProviderID | ID of the service provider account the application is associated with.  
**Note:** If left empty, the mapping will *not* be applied on service provider account and application account level. |
| applicationID     | ID of the application account.  
**Note:** If left empty, the mapping will *not* be applied on application account level.                                                   |
| serviceType       | OSA/Parlay service type name (TpServiceTypeName) of the OSA/Parlay SCS to which the OSA Client is to be mapped.  
See the specification for the OSA/Parlay Framework for a list of recommended service type names. |
| osaClientAppId    | The OSA/Parlay account’s clientAppID, a string consisting of the entOpId followed by \, followed by the appId. Example: sp1\app1.  
The entOpId and appId is provided by the OSA Gateway administrator. |
| properties        | OSA/Parlay service properties to be used in the look up (service discovery) phase when requesting a service (OSA/Parlay SCS) from the OSA/Parlay Gateway.  
The properties are specified as a space separated list in the following way:  
<propname1> <propval1> <propname2> <propval2>  
The properties varies between OSA/Parlay Gateway implementations. |
| authType          | Authentication type to be used. The type is defined according to the OSA/Parlay standard. P_AUTHENTICATION is the only supported.  
**Note:** When P_AUTHENTICATION is used, no encryption or signing algorithm will be used and the parameters encryptionMethod and signingAlgorithm can be left empty. |
| encryptionMethod  | Method used for encryption. The type is defined according to OSA/Parlay standard. If the type is not specified, enter P_RSA_1024. |
Managing and Configuring OSA/Parlay Gateway Connections

Table 27-4  AddMapping

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>signingAlgorithm</td>
<td>Signing algorithm. The type is defined according to OSA/Parlay standard. If the type is not specified, enter P_MD5_RSA_1024.</td>
</tr>
<tr>
<td>gatewayId</td>
<td>OSA/Parlay Gateway ID. This ID was generated when the OSA/Parlay Gateway was created, see Attribute: EricssonAuthentication, and Operation: listActiveMappings.</td>
</tr>
<tr>
<td>initConnection</td>
<td>Indicating if the connection to OSA/Parlay Gateway should be initialized immediately. That is, if authentication should performed when the Operation: addClient operation is invoked.</td>
</tr>
</tbody>
</table>

**Operation: listActiveMappings**

Scope: Server

Lists the IDs active OSA Client Mappings.

Signature:

listActiveMappings()

Table 27-5  listActiveMappingsForGw

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

Return List of IDs for active mappings.

**Operation: listActiveMappingsForGw**

Scope: Server
Lists the IDs of all active OSA Client Mappings for a specific OSA Gateway.

Signature:

`listActiveMappingsForGw(gwId: int)`

Table 27-6  listActiveMappingsForGw

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gwId</td>
<td>The ID of the OSA Gateway.</td>
</tr>
</tbody>
</table>

Return: List of IDs for active mappings.

**Operation: listGw**

Scope: Server

Lists the IDs of all registered OSA Gateways.

Signature:

`listGw()`

Table 27-7  listGw

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Return: List of registered OSA Gateways.

**Operation: listMappings**

Scope: Server
Lists the configured OSA Client Mappings.

Signature:
listMappings()

Table 27-8 listMappings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Return: List of OSA Client mappings.

**Operation: removeClient**

Scope: Server

Removes an OSA client.

Signature:
removeClient(osaClientAppId: String, keystorePwd: String)

Table 27-9 removeClient

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>osaClientAppId</td>
<td>The OSA client application ID (and alias in keystore). See Operation: addClient.</td>
</tr>
<tr>
<td>keystorePwd</td>
<td>Network Gatekeeper keystore password.</td>
</tr>
</tbody>
</table>

**Operation: removeConnection**

Scope: Server
Removes an OSA Gateway Connection

Signature:

removeConnection(gatewayId: int, connectionId: int)

Table 27-10  removeConnection

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gatewayId</td>
<td>The ID of the OSA Gateway.</td>
</tr>
<tr>
<td>connectionId</td>
<td>The ID of the connection. The ID was returned when the connection was setup, see Operation: addConnection and Operation: listActiveMappings</td>
</tr>
</tbody>
</table>

**Operation: removeGw**

Scope: Server

Removes an OSA Gateway.

Signature:

removeGw(id: int, keystorePwd: String)

Table 27-11  removeGw

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the OSA Gateway to remove. The ID was returned when the OSA Gateway was created, see Operation: addGw and Operation: listGw</td>
</tr>
<tr>
<td>keystorePwd</td>
<td>The Network Gatekeeper keystore password.</td>
</tr>
</tbody>
</table>

**Operation: removeMapping**

Scope: Server

Removes a client mapping.
Managing and Configuring OSA/Parlay Gateway Connections

Signature:

removeMapping(id: int)

Parameters:

Table 27-12 removeMapping

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>ID of the OSA Client Mapping to remove.</td>
</tr>
</tbody>
</table>

Operation: setKeyStorePassword

Scope: Server

Sets the password that protects the keystore.

Signature:

setKeyStorePassword(newPassword: String, oldPassword: String)

Table 27-13 setKeyStorePassword

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>newPassword</td>
<td>The new password for the keystore.</td>
</tr>
<tr>
<td>oldPassword</td>
<td>The old password for the keystore.</td>
</tr>
</tbody>
</table>

Operation: viewActiveMappingState

Scope: Server

Displays the state of an active mapping OSA Client Mapping.

Signature:

viewActiveMappingState(mappingId: int)
Table 27-14 viewActiveMappingState

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mappingId</td>
<td>The ID of the OSA Client Mapping.</td>
</tr>
</tbody>
</table>

Return: List of configured mappings.
Callable Policy Web Service

The following section describes how to use the callable policy interface exposed by Network Gatekeeper.

- Introduction
- Callable Policy Web Service interface definition
  - Endpoints
  - Detailed service description
  - XML Schema data type definition
    - AdditionalDataValue structure
    - AdditionalDataValueType enumeration
  - Interface: Policy
    - Operation: evaluate
  - Interface: PolicyManagement
    - Operation: reloadXmlDriver
    - Operation: viewRuleFile
    - Operation: deleteRuleFile
    - Operation: loadRules
    - Operation: listRuleFiles
Callable Policy Web Service

- Rule files

Introduction

The callable policy service in WebLogic Network Gatekeeper exposes two Web Services interfaces related to callable policy:

- Policy evaluation
- Policy management

The callable policy service is intended to allow applications and network nodes that have no policy evaluation capabilities themselves to use the policy evaluation capabilities in Network Gatekeeper. Although the callable policy service is implemented in the same manner as a traffic path, it is not designed to expose the service to external service providers. Rather it is to be used internally as a way of exposing generic policy capabilities to network nodes within the telecom network where Network Gatekeeper is deployed. Traffic paths and network protocol plug-ins deployed in Network Gatekeeper do not use the interfaces exposed by the callable policy Web Service.

For example, a node in the network might need to enforce a set of rules for requests flowing through it, to allow or deny requests based on time of day and originator of the request. In this case, the node might determine the originator of the request and use the callable policy evaluation Web Service to evaluate that request. The rule that is being evaluated uses the data provided in the web services call and makes its decision based on them. Modifications to the rules can be done using the policy management Web Service.

A user of the policy evaluation and policy management Web Services interfaces is registered using the same service provider and application model that is used for users of the exposed traffic paths, and it must be logged in using the same session manager interface exposed to these service provider applications.

Note: If there is no specific rule file associated with a ServiceName loaded in the rule engine, it uses the default rule file in its evaluation. Because the default rule is designed to enforce Service Level Agreements, this is not appropriate for use with the Callable Policy Web Service. If you are using Callable Policy, you must make sure that an appropriate rule file is loaded into the rule engine. For more information, see Operation: loadRules.

It necessary to have service provider group and application group Service Level Agreements defined for the user of the callable policy service. To use the policy evaluation interface, the tag <scs> must contain the value com.bea.wlcp.wlnt.png.px21.plugin.PolicyPlugin.
Callable Policy Web Service interface definition

To use the policy management interface, the tag `<scs>` must contain the value com.bea.wlcp.wlng.px21.plugin.PolicyManagementPlugin.

```
Listing 28-1   Example of SLA that allows the use of both the policy evaluation and policy management interfaces

<serviceContract>
  <scs>com.bea.wlcp.wlng.px21.plugin.PolicyPlugin</scs>
</serviceContract>
<serviceContract>
  <scs>com.bea.wlcp.wlng.px21.plugin.PolicyManagementPlugin</scs>
</serviceContract>
```

Callable Policy Web Service interface definition

Endpoints

The endpoint for the Policy evaluation interface is:
http://<host:port>/callable_policy/Policy

The endpoint for the Policy management interface is:
http://<host:port>/callable_policy/PolicyManagement

Detailed service description

Policy Evaluation

The policy evaluation interface makes it possible for an external application to evaluate a request containing a set of parameters. The parameters in the request include authentication information, information on the type of service the request should be evaluated against, the method name of the method that should be evaluated, and arbitrary additional data provided as name-value pairs. All request parameters are evaluated according to a policy rule.
When evaluated, a copy of the data provided in the evaluation process is returned together with information on the outcome of the requests, that is, if the request was allowed or denied. If the request was allowed, the application calling the Web Service must use the returned copy of the parameters for further processing, because the returned parameters in the request may have been changed by the policy rule processing.

**Policy management**

The policy management web service interface makes it possible to load and delete policy rules.

**XML Schema data type definition**

**AdditionalDataValue structure**

Defines the AdditionalDataValue structure.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Element type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>xsd:string</td>
<td>N</td>
<td>Name part of the additional data name-value pair.</td>
</tr>
<tr>
<td>value</td>
<td>xsd:string</td>
<td>N</td>
<td>Value part of the additional data name-value pair.</td>
</tr>
<tr>
<td>type</td>
<td>callable_policy_local_xsd:AdditionalDataValueType</td>
<td>N</td>
<td>Identifies the data type. See AdditionalDataValueType enumeration.</td>
</tr>
</tbody>
</table>

**AdditionalDataValueType enumeration**

Describes a data type.

<table>
<thead>
<tr>
<th>Enumeration value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING_TYPE</td>
<td>Data type is String.</td>
</tr>
<tr>
<td>INTEGER_TYPE</td>
<td>Data type is Integer.</td>
</tr>
</tbody>
</table>
Callable Policy Web Service interface definition

Interface: Policy
Operations to evaluate a request.

Operation: evaluate
The policy evaluation interface makes it possible for an external application to evaluate a request containing a set of parameters. All of the request parameters are evaluated according to a Policy rule.

Input message: evaluateRequest

<table>
<thead>
<tr>
<th>Part name</th>
<th>Part type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>xsd:string</td>
<td>N</td>
<td>Service type to be evaluated.</td>
</tr>
<tr>
<td>serviceName</td>
<td>xsd:string</td>
<td>N</td>
<td>ServiceName associated with the rule file.</td>
</tr>
<tr>
<td>methodName</td>
<td>xsd:string</td>
<td>N</td>
<td>Name of method to be evaluated.</td>
</tr>
<tr>
<td>requesterID</td>
<td>xsd:string</td>
<td>N</td>
<td>The application ID as given by the operator.</td>
</tr>
</tbody>
</table>
Callable Policy Web Service

### Output message: evaluateResponse

<table>
<thead>
<tr>
<th>Part name</th>
<th>Part type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeStamp</td>
<td>xsd:dateTime</td>
<td>N</td>
<td>Defines the date and time of the request.</td>
</tr>
<tr>
<td>additionalData</td>
<td>callable_polic y_local_xsd:additionalDataValue</td>
<td>Y</td>
<td>Specifies any other data, specified as name-value pairs. See AdditionalDataValue structure.</td>
</tr>
<tr>
<td>modifiedRequest</td>
<td>callable_polic y_local_xsd:evaluateRequest</td>
<td>N</td>
<td>The response that Network Gatekeeper returns after being evaluated by policy rules. Same data structure as evaluateRequest, but data may have been changed by the policy evaluation.</td>
</tr>
<tr>
<td>returnValue</td>
<td>xsd:string</td>
<td>N</td>
<td>Return value the policy rules passed back.</td>
</tr>
<tr>
<td>thrownException</td>
<td>xsd:string</td>
<td>N</td>
<td>Name of the exception thrown during evaluation.</td>
</tr>
<tr>
<td>thrownPolicyException</td>
<td>xsd:string</td>
<td>N</td>
<td>Name of the policy rejection exception thrown during evaluation.</td>
</tr>
<tr>
<td>denyReason</td>
<td>xsd:string</td>
<td>N</td>
<td>Description of the reason of denying the request.</td>
</tr>
<tr>
<td>denyCode</td>
<td>xsd:string</td>
<td>N</td>
<td>Code identifying the reason of denying the request.</td>
</tr>
</tbody>
</table>

### Referenced faults

**ServiceException:**

If there is an internal error during evaluation process, a ServiceException is thrown.

**PolicyException:**

If the policy evaluation request is rejected, a PolicyException is thrown.
**Interface: PolicyManagement**
Operations to manage policy rules.

**Operation: reloadXmlDriver**
Reloads an XML file for SLAs into the rules engine.

**Input message: reloadXmlDriver**

<table>
<thead>
<tr>
<th>Part name</th>
<th>Part type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>xsd:string</td>
<td>N</td>
<td>Type of SLA, either:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Application</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• ServiceProvider</td>
</tr>
</tbody>
</table>

**Output message: reloadXmlDriverResponse**

<table>
<thead>
<tr>
<th>Part name</th>
<th>Part type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>.</td>
</tr>
</tbody>
</table>

**Referenced faults**

ServiceException:
If there is an internal error during evaluation process, a ServiceException is thrown.

PolicyException:
If the policy evaluation request is rejected, a PolicyException is thrown.

**Operation: viewRuleFile**
Fetches a policy rule file of a given type and service from the rules engine.
Callable Policy Web Service

**Input message: viewRuleFile**

<table>
<thead>
<tr>
<th>Part name</th>
<th>Part type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>xsd:string</td>
<td>N</td>
<td>Type of SLA, either:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Application</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• ServiceProvider</td>
</tr>
<tr>
<td>serviceName</td>
<td>xsd:string</td>
<td>N</td>
<td>ServiceName associated with the rule file.</td>
</tr>
</tbody>
</table>

**Output message: viewRuleFileResponse**

<table>
<thead>
<tr>
<th>Part name</th>
<th>Part type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>return</td>
<td>xsd:string</td>
<td>N</td>
<td>The rule file.</td>
</tr>
</tbody>
</table>

**Referenced faults**

ServiceException:
If there is an internal error during evaluation process, a ServiceException is thrown.

PolicyException:
If the policy evaluation request is rejected, a PolicyException is thrown.

**Operation: deleteRuleFile**
Deletes a policy rule file of a given type and service from the rules engine.
Callable Policy Web Service interface definition

Input message: deleteRuleFile

<table>
<thead>
<tr>
<th>Part name</th>
<th>Part type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>xsd:string</td>
<td>N</td>
<td>Type of rule file, either:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Application</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• ServiceProvider</td>
</tr>
<tr>
<td>serviceName</td>
<td>xsd:String</td>
<td>N</td>
<td>ServiceName associated with the rule file.</td>
</tr>
</tbody>
</table>

Output message: deleteRuleFileResponse

Referenced faults

ServiceException:
If there is an internal error during evaluation process, a ServiceException is thrown.

PolicyException:
If the policy evaluation request is rejected, a PolicyException is thrown.

Operation: loadRules
Loads a policy rule file of a given type and service into the rules engine.
Callable Policy Web Service

Input message: loadRules

<table>
<thead>
<tr>
<th>Part name</th>
<th>Part type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>xsd:string</td>
<td>N</td>
<td>Type of rule file, either:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Application</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Serviceprovider</td>
</tr>
<tr>
<td>irlUrl</td>
<td>xsd:string</td>
<td>N</td>
<td>URL to rule file to be loaded.</td>
</tr>
<tr>
<td>serviceName</td>
<td>xsd:string</td>
<td>N</td>
<td>ServiceName associated with the rule file.</td>
</tr>
</tbody>
</table>

Output message: loadRulesResponse

<table>
<thead>
<tr>
<th>Part name</th>
<th>Part type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
</table>

Referenced faults

ServiceException:
If there is an internal error during evaluation process, a ServiceException is thrown.

PolicyException:
If the policy evaluation request is rejected, a PolicyException is thrown.

Operation: listRuleFiles
Lists the rule files of a given type that are loaded into the rules engine.
Input message: listRuleFiles

<table>
<thead>
<tr>
<th>Part name</th>
<th>Part type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>xsd:string</td>
<td>N</td>
<td>Type of rule file, either:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Application</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• ServiceProvider</td>
</tr>
</tbody>
</table>

Output message: listRuleFilesResponse

<table>
<thead>
<tr>
<th>Part name</th>
<th>Part type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ruleFile</td>
<td>Array of xsd:string</td>
<td>Y</td>
<td>A list of rule files matching the given criteria.</td>
</tr>
</tbody>
</table>

Referenced faults

ServiceException:
If there is an internal error during evaluation process, a ServiceException is thrown.

PolicyException:
If the policy evaluation request is rejected, a PolicyException is thrown.

Rule files

The rule files are written in IRL, ILog Rule Language.
When writing rules in the context of Network Gatekeeper policy rules, the following apply:
The rule is associated with a service name when loaded into Network Gatekeeper policy service, Input message: loadRules.
Which rule to be triggered by Input message: evaluateRequest is correlated with the parameter serviceName given in the Web Service request.
When the evaluate request triggers the rule, a set of general parameters can be accessed by the policy rule:
Callable Policy Web Service

- **String applicationID**: Application ID associated with the request.
- **String serviceProviderID**: Service provider ID associated with the request.
- **String serviceName**: Service name from which the request originates or is destined for.
- **String methodName**: Method that triggered the request.
- **String serviceCode**.
- **String requesterID**.
- **long transactionID**.
- **int noOfActiveSessions**.
- **long timeStamp**: Time the request was sent to the rules engine for processing. Milliseconds from start of UNIX epoch.
- **long reqCounter**: Defines the increase rate for related counters.

A rule must have a name and a priority. High priority rules are evaluated before low priority rules. There are a set of pre-defined priority levels, which are mapped to a numerical value:

- **minimum**, where the value is -1*10^9
- **low**, where the value is -1*10^6
- **high**, where the value is 1*10^6
- **maximum**, where the value is 1*10^9

Listing 16-2 shows the basic structure of a rule:

**Listing 28-2  Skeleton of a rule**

```java
rule DenySubscriberNotExists
{
    priority = high;
    when
    { // fetch the policy request data and perform evaluations.
    }
}
```

28-12 System Administrator's Guide
In order to perform an evaluation, the data in the PolicyRequest object must be fetched by the rule and mapped to the equivalent variable names in the rule. The standard types of request data in the Policy Request are associated with variables of the same name in the rules. Below is an example of a rule assigning the PolicyRequest member variable serviceName to the rule variable sname via the Policy Request object. The rule object pr is assigned to the PolicyRequest object.

**Listing 28-3  Policy Request data is fetched**

```.
?pr: event PolicyRequest(?sname: serviceName);
```

If the Policy Engine has evaluated the request and made the decision to deny it, the Policy Engine’s representation of the PolicyRequest object (pr) must be retracted. Retracting the PolicyRequest object aborts further rule enforcement.

**Listing 28-4  Retract a request**

```.
retract (?pr);
```

If the Policy Engine has evaluated the request and made the decision to allow it, the Policy Engine’s representation of the request (pr) must still be retracted, but in the last rule of the execution flow. For example, this could be achieved by adding a general finalizing allow rule that retracts the request. This rule should have priority minimum.
Listing 28-5  General finalizing allow rule that retracts a request

rule AllowServiceRequest
{
    priority = minimum;
    when
    {
        ?pr: event PolicyRequest();
    }
    then
    {
        retract (?pr);
        ?pr.allow();
    }
}

Data that is defined as AdditionalValues must fetched as shown below. The Additional Value named targetAddress is stored in the variable addDataValue. The PolicyRequest object is pr.

Listing 28-6  Fetching AdditionalValue data

bind ?addDataValue = ?pr.getAdditionalDataStringValue("targetAddress");

The particular signature of the fetching method depends on the type of data:

- getAdditionalDataIntValue(...), for int values
- getAdditionalDataLongValue(...), for long value.
• `getAdditionalDataStringValue(...)`, for String values
• `getAdditionalDataStringArrayValue(...)`, for arrays of String values
• `getAdditionalDataBooleanValue(...)`, for boolean values
• `getAdditionalDataShortValue(...)`, for short values
• `getAdditionalDataCharValue(...)`, for char values
• `getAdditionalDataFloatValue(...)`, for float values
• `getAdditionalDataDoubleValue(...)`, for double values
• `getAdditionalDataIntArrayValue(...)`, for arrays of int values.

If the data type is unknown, it can be determined by invoking the discriminator method on the `AdditionalDataValue` object.

**Listing 28-7  Determine the type of an AdditionalDataValue**

```java
bind ?type = ?pr.getAdditionalData.dataValue.discriminator().value();
```

Where `type` is one of the following:

- `AdditionalDataType._P_ADDITIONAL_INT`
- `AdditionalDataType._P_ADDITIONAL_LONG`
- `AdditionalDataType._P_ADDITIONAL_SHORT`
- `AdditionalDataType._P_ADDITIONAL_STRING`
- `AdditionalDataType._P_ADDITIONAL_STRING_ARRAY`
- `AdditionalDataType._P_ADDITIONAL_BOOLEAN`
- `AdditionalDataType._P_ADDITIONAL_FLOAT`
- `AdditionalDataType._P_ADDITIONAL_DOUBLE`
- `AdditionalDataType._P_ADDITIONAL_INT_ARRAY`
Managing the Policy Service

The following section describes how to manage the Policy service:

- Introduction
- Configuration workflow
- Reference: attributes and operations for Policy

Introduction

The Policy service evaluates requests according to rules and data. The data come from two sources:

- Service level agreements
- Data provided in requests to an application.

Management of the Policy service mainly involves loading and deleting the rules that enforce the SLAs and the request data. If a traffic path has no rule files configured, the DEFAULT rule file is used.

Configuration workflow

No configuration is applicable.
Reference: attributes and operations for Policy

Below is a list of attributes and operations for configuration and maintenance.

- Operation: deleteApplicationRuleFile
- Operation: deleteServiceProviderRuleFile
- Operation: listApplicationRuleFiles
- Operation: listServiceProviderRuleFiles
- Operation: loadApplicationRules
- Operation: loadServiceProviderRules
- Operation: reloadApplicationXmlDriver
- Operation: reloadNodeXmlDriver
- Operation: reloadServiceProviderXmlDriver
- Operation: viewApplicationRuleFile
- Operation: viewServiceProviderRuleFile

**Operation: deleteApplicationRuleFile**

Deletes a rule file on the application group level.

Signature:

deleteApplicationRuleFile(serviceName: String)

<table>
<thead>
<tr>
<th>Table 29-1 deleteApplicationRuleFile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>serviceName</td>
</tr>
</tbody>
</table>
**Operation: deleteServiceProviderRuleFile**

Deletes a rule file on the service provider group level.

Signature:

```
deleteServiceProviderRuleFile(serviceName: String)
```

**Table 29-2  deleteServiceProviderRuleFile**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>Name of the service for which the rule file is applied.</td>
</tr>
</tbody>
</table>

**Operation: listApplicationRuleFiles**

Displays all application group rule files.

Signature:

```
listApplicationRuleFiles()
```

**Table 29-3  listApplicationRuleFiles**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Operation: listServiceProviderRuleFiles**

Displays all service provider group rule files.

Signature:

```
listServiceProviderRuleFiles()
```
Managing the Policy Service

Table 29-4  listServiceProviderRuleFiles

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Operation: loadApplicationRules**

Loads a rule file on the application group level.

Signature:

\[ \text{loadApplicationRules(serviceName: String)} \]

Table 29-5  loadApplicationRules

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>irlUrl</td>
<td>URL to the rule file.</td>
</tr>
<tr>
<td>serviceName</td>
<td>Name of the service for which the rule file is applied.</td>
</tr>
</tbody>
</table>

**Operation: loadServiceProviderRules**

Loads a rule file on the service provider group level.

Signature:

\[ \text{loadServiceProviderRules(serviceName: String)} \]
Operation: reloadApplicationXmlDriver

Reloads an XML driver. Necessary if the XML schema file has been changed for an application group level SLA.

Signature:

reloadApplicationXmlDriver()

Table 29-6  loadServiceProviderRules

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>irlUrl</td>
<td>URL to the rule file.</td>
</tr>
<tr>
<td>serviceName</td>
<td>Name of the service for which the rule file is applied.</td>
</tr>
</tbody>
</table>

Operation: reloadNodeXmlDriver

Not applicable.

Operation: reloadServiceProviderXmlDriver

Reloads an XML driver. Necessary if the XML schema file has been changed for a service provider group level SLA.

Signature:

reloadServiceProviderXmlDriver()
Managing the Policy Service

Table 29-8 reloadServiceProviderXmlDriver

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Operation: viewApplicationRuleFile

Displays the contents of a rule file on application group level.

Signature:

```
viewApplicationRuleFile(serviceName: String)
```

Table 29-9 viewApplicationRuleFile

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>Name of the service for which the rule file is applied.</td>
</tr>
</tbody>
</table>

Operation: viewServiceProviderRuleFile

Displays the contents of a rule file on service provider group level.

Signature:

```
viewServiceProviderRuleFile(serviceName: String)
```

Table 29-10 viewServiceProviderRuleFile

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceName</td>
<td>Name of the service for which the rule file is applied.</td>
</tr>
</tbody>
</table>
Writing OAM Scripts and using JMX

The following section gives examples of how to use WebLogic Scripting Tool (WLST) to write scripts or how to use WLST in interactive mode when configuring and managing Network Gatekeeper. It serves as an introduction both to using WLST to manage Network Gatekeeper interactively and to using Python scripts for scripted management of Network Gatekeeper.

For information about Weblogic Server and WLST, see http://edocs.bea.com/wls/docs92/config_scripting/using_WLST.html

Note: A full set of JavaDoc for all OAM attributes and operations can be found by going to the Reference or Integration topic pages in the password protected section of the Network Gatekeeper 3.0 documentation pages on edocs.bea.com.

- Working in interactive mode
  - Starting WLST and connecting to Network Gatekeeper
  - Exiting WLST
  - Changing an attribute
  - Invoking an operation
- Scripting WLST
- Using JMX for Management
Working in interactive mode

Starting WLST and connecting to Network Gatekeeper

1. Make sure the correct Java environment is set:
   
   UNIX: `$Domain_Home/bin/setDomainEnv.sh`
   or
   Windows: `$Domain_Home\bin\setDomainEnv.cmd`

2. Start WLST: `java weblogic.WLST`

3. Connect to the server to manage:
   
   ```
   connect('<username>','<password>','t3://<host>:<port>')
   ```

4. Change to custom tree where Network Gatekeeper MBeans are located:
   
   ```
   custom()
   cd('com.bea.wlcp.wlng')
   ```

5. To display a list of the MBeans: `dir()`

   The MBean names are also displayed in each Configuration and Provisioning page for the
   management objects in the Network Gatekeeper Management Console.

6. Select the MBean to change:
   
   ```
   cd('com.bea.wlcp.wlng:Name=wlng_nt,Type=<MBean name>')
   ```

   For example, to select the MBean for the EDRService, use:
   
   ```
   cd('com.bea.wlcp.wlng:Name=wlng_nt,Type=com.bea.wlcp.wlng.edr.management.EdrServiceMBean')
   ```

Exiting WLST

1. Disconnect from Network Gatekeeper: `disconnect()`

2. Exit the WLST shell: `exit()`

Changing an attribute

1. Select the MBean to change attribute for.
2. Set the attribute:
   ```python
   set('<name of attribute>', <value of attribute>)
   ```

3. For example, to change the attribute `BatchSize` to `2001` in the managed object `EDRSservice`:
   ```python
   set('BatchSize', 2001)
   ```

4. The attribute values for an MBean can be displayed using: `ls()`

## Invoking an operation

1. Select the MBean to invoke an operation on.

2. Define the parameters as an array:
   ```python
   objs = jarray.array([java.lang.String('<value>'),
                        java.lang.String('<value>')], java.lang.Object)
   ```

3. Define the data types as an array:
   ```python
   strs = jarray.array(['java.lang.String',
                       'java.lang.String'], java.lang.String)
   ```

4. Invoke the operation: `print(invoke('<name of method>',objs,strs))`

   For example, to invoke the `displayStatistics` method in the `EDRSservice` MBean:
   ```python
   cd('com.bea.wlcp.wlng:Name=wlng_nt,Type=com.bea.wlcp.wlng.edr.management.EdrServiceMBean')
   objs = jarray.array([], java.lang.Object)
   strs = jarray.array([], java.lang.String)
   print(invoke('displayStatistics',objs,strs))
   ```

   This operation has no arguments.

   Another example is to add a route using the `Plugin_Manager` MBean:
   ```python
   cd('com.bea.wlcp.wlng:Name=wlng_nt,Type=com.bea.wlcp.wlng.management.Plugin_managerMBean')
   objs = jarray.array([java.lang.String('Plugin_presence_sip'),
                        java.lang.String('sip:*')], java.lang.Object)
   strs = jarray.array(['java.lang.String',
                        'java.lang.String'], java.lang.String)
   print(invoke('addRoute',objs,strs))
   ```

   The method `addRoute` takes two arguments, and the values of the parameters are stated in the same order as the parameters are defined in the signature of the method.
Note: For a method that has input parameters and a string return value, invoke command can be executed as follows:

```java
objs = jarray.array([java.lang.String('stringInput1'), java.lang.String('stringInput2'), java.lang.String('stringInput3')], java.lang.Object)
invoke('methodName', objs, strs)
```

### Scripting WLST

When using WLST together with scripts, WLST is invoked according to:

```java
java weblogic.WLST <script name>.py <argument 1> <argument 2> ...<argument n>
```

The arguments can be retrieved from within the scripts in the array `sys.argv[]`, where `sys.argv[0]` is the script name, `sys.argv[1]` is the second argument and so on. It may be useful to specify login information and connection information as arguments to the scripts.

Below is a script that connects to Network Gatekeeper and changes to an MBean defined in by argument.

### Listing 30-1 Example of script

```python
userName = sys.argv[1]
passWord = sys.argv[2]
objectType = sys.argv[5]
objectName = "com.bea.wlcp.wlng:Name=wlng_nt,Type=" + sys.argv[5]
print objectName
connect(userName, passWord, url)
custom()
cd('com.bea.wlcp.wlng')
cd(objectName)
```
Example on how to invoke the script:

```
java weblogic.WLST script1.py weblogic weblogic localhost 7001
com.bea.wlcp.wlng.edr.management.EdrServiceMBean
```

**Using JMX for Management**

Network Gatekeeper exposes its management interfaces as JMX MBeans. These MBeans come in two flavours:

- Standard MBeans
- Dynamic MBeans

The standard MBeans are exposed by all core services, core utilities and plug-ins new to Network Gatekeeper 3.0, while the dynamic MBeans are used for the services that were kept from Network Gatekeeper 2.2. The dynamic MBeans are generated in runtime based on the OAM interfaces used in network Gatekeeper.

JavaDoc for the Management interfaces are found in [JavaDoc for Management Operations](#). The dynamic MBeans are identified in the JavaDoc by the name of the interface. The names always have the suffix `OAMOperations`. The JavaDoc for these differs slightly from the interface itself:

- When the JavaDoc states that a non-primitive data type is returned, the actual return value is an array of Strings.

- When the JavaDoc states that a non-primitive data type is used as an in-parameter, the actual parameters are the same as displayed in the Configuration and Provisioning page for the service in the Network Gatekeeper administration console.

- The ObjectName to use is displayed in the Configuration and Provisioning page for the service in the Network Gatekeeper administration console.

Below is an example on how to connect to the MBean server and perform an operation on the JMX interfaces.
Listing 30-2  Example of using JMX to manage Network Gatekeeper

```java
import java.io.IOException;
import java.net.MalformedURLException;
import java.util.Hashtable;

import javax.management.MBeanServerConnection;
import javax.management.MalformedObjectNameException;
import javax.management.ObjectName;
import javax.management.remote.JMXConnector;
import javax.management.remote.JMXConnectorFactory;
import javax.management.remote.JMXServiceURL;
import javax.naming.Context;

public class TestMgmt {
    private static MBeanServerConnection connection;
    private static JMXConnector connector;

    /*
     * Initialize connection to the Domain Runtime MBean Server
     */
    public static void initConnection(String hostname, String portString,
                                      String username, String password) throws
                                      IOException, MalformedURLException {
        String protocol = "t3";
    }
```
Integer portInteger = Integer.valueOf(portString);
int port = portInteger.intValue();
String jndiroot = "\jndi/";
String mserver = "weblogic.management.mbeanservers.domainruntime";
JMXServiceURL serviceURL = new JMXServiceURL(protocol, hostname,
    port, jndiroot + mserver);
Hashtable h = new Hashtable();
h.put(Context.SECURITY_PRINCIPAL, username);
h.put(Context.SECURITY_CREDENTIALS, password);
h.put(JMXConnectorFactory.PROTOCOL_PROVIDER_PACKAGES,
    "weblogic.management.remote");
connector = JMXConnectorFactory.connect(serviceURL, h);
connection = connector.getMBeanServerConnection();
}

public static void main(String[] args) throws Exception {
    String hostname = args[0]; //hostname of the admin server
    String portString = args[1]; //port of the admin server
    String username = args[2];
    String password = args[3];
    String mbserverName = args[4]; //NT server name

    String operationName = "removeShortCodeTranslation";
    String originalDestinationAddressPattern = "";
    String translatedDestinationAddress = "";
    String[] params = new String[] {originalDestinationAddressPattern,
        translatedDestinationAddress};
Writing OAM Scripts and using JMX

String[] signature = new String[] {"java.lang.String", "java.lang.String"};
ObjectName on;

try {
    on = new ObjectName("com.bea.wlcp.wlng:Name=wlng_nt,Type=com.bea.wlcp.wlng.plugin.sms.smpp.management.SmsMBean,Location=" + mbserverName);
} catch (MalformedObjectNameException e) {
    throw new AssertionError(e.getMessage());
}

initConnection(hostname, portString, username, password);

//invoke the operation

Object result = connection.invoke(on, operationName, params, signature);
System.out.println(result.toString());  //displays the result
connector.close();
Creating an EDR listener

The following section describes how to create an external EDR listener.

- **Overview of External EDR listeners**
  - Example using a pure JMS listener
  - Example using JMSListener utility with no filter
  - Example using JMSListener utility with no filter

- **Description of EDR listener utility**
  - Class JMSListener
  - Class EdrFilterFactory
  - Class EdrData
  - Class ConfigDescriptor
  - Class EdrConfigDescriptor
  - Class AlarmConfigDescriptor
  - Class CdrConfigDescriptor

**Overview of External EDR listeners**

External EDR listeners are JMS topic subscribers.

The diagram below illustrates three different ways of listening for EDRs as a JMS listener.
EDRs are published externally using a JMS topic which makes it possible to implement language-independent listeners anywhere on the network in a standard way. It is possible to implement an EDR listener in several ways:

- **Alternative 1:** Using a pure JMS listener. Implement the javax.jms.MessageListener interface. It is up to the implementation class to implement any filtering mechanism needed.

- **Alternative 2:** Using a subclass of JMSListener with no filter specified. In that case, the JMSListener class will use a tag, if available in the EDR, to filter the EDR in a specific category: EDR, alarm or CDR.
Overview of External EDR listeners

- Alternative 3: Using a subclass of JMSListener with a specified filter. This filter is used to perform the filtering. If a default filter is used to perform the same filtering as the WLNG, note that all classes used in the xml configuration files must be present in the current class loader. Otherwise, some EDRs will not be correctly filtered.

**Example using a pure JMS listener**

Listing 31-1  Example using a pure JMS listener

```java
public class ClientJMSListener implements MessageListener {
    public void onMessage(Message msg) {
        // Extract the EdrData object or array
        if(o instanceof EdrData[]) {
            for(EdrData edr : (EdrData[])o) {
                //do something with each EDR
            }
        }
    }
}
```

**Example using JMSListener utility with no filter**

Listing 31-2  Example using a subclass of JMSListener with no filter specified

```java
public class SampleEdrJMSListener extends JMSListener {
    public SampleEdrJMSListener(String url) throws Exception {
        // Register in the JMS topic. No filter is specified so
        // the "tag" filtering mechanism will be used.
        register(url);
    }
```
Creating an EDR listener

```java
@Override
public void onEdr(EdrData edr, ConfigDescriptor descriptor) {
    // The "tag" mechanism will filter the stream of EDRs according
    // to the internal WLNG filtering. To know which type of EDR is
    // actually provided in this method, we have to determine the
    // instance of the ConfigDescriptor as follow:
    if(descriptor instanceof EdrConfigDescriptor) {
        // do something with this EDR
    } else if(descriptor instanceof AlarmConfigDescriptor) {
        // do something with this alarm
    } else if(descriptor instanceof CdrConfigDescriptor) {
        // do something with this CDR
    }
}
```

Using JMSListener utility with a filter

Listing 31-3 Using a subclass of JMSListener with a specified filter

```java
public class SampleEdrJMSListener extends JMSListener {

    public SampleEdrJMSListener(String url) throws Exception {
        // Register in the JMS topic. Use the default alarm filter.
        // Note that in this case all classes needed by the alarm.xml file
        // must be in the current class loader in order for the filtering
```

Description of EDR listener utility

// to work correctly.
register(url, EdrFilterFactory.createDefaultFilterForAlarm());
}

@Override
public void onEdr(EdrData edr, ConfigDescriptor descriptor) {
    // Only AlarmConfigDescriptor should be received here.
    // Just check before casting.
    if(descriptor instanceof AlarmConfigDescriptor) {
        ... do something with this alarm
    }
}

Note: When using the JMSListener class, make sure that any modification to an xml configuration file in the WLNG is also updated in the edrjmslistener.jar file.

Description of EDR listener utility

The EDR listener utility contains a set of classes to use when creating an external JMS listener using the JMSListener.

The helper classes are found in the domain home directory in Network Gatekeeper, in:
<domain home>/edr/edrjmslistener.jar
Creating an EDR listener

Class JMSListener

Table 31-1  JMSListener

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public void register(String url)</td>
<td>Registers the JMS listener to the EDR topic using no filter. The filtering will be done using the tagging mechanism. The parameter url specifies the URL of a network tier WLNG server.</td>
</tr>
<tr>
<td>public void register(String url, EdrFilter filter)</td>
<td>Registers the JMS listener to the EDR topic using the specified filter.</td>
</tr>
<tr>
<td>public void onEdr(EdrData edr, ConfigDescriptor descriptor)</td>
<td>Method that the subclass can override to get notified each time an EDR is received. The descriptor will be a subclass of ConfigDescriptor that will identify the type of EDR: either EdrConfigDescriptor, AlarmConfigDescriptor or CdrConfigDescriptor.</td>
</tr>
</tbody>
</table>

Class EdrFilterFactory

Table 31-2  EdrFilterFactory

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static EdrFilter createDefaultFilterForEdr()</td>
<td>Creates the default filter using in the WLNG to filter the EDRs using the edr.xml file embedded in the edrjmslistener.jar file.</td>
</tr>
<tr>
<td>public static EdrFilter createDefaultFilterForAlarm()</td>
<td>Creates the default filter using in the WLNG to filter the alarms using the alarm.xml file embedded in the edrjmslistener.jar file.</td>
</tr>
<tr>
<td>public static EdrFilter createDefaultFilterForCdr()</td>
<td>Creates the default filter using in the WLNG to filter the CDRs using the cdr.xml file embedded in the edrjmslistener.jar file.</td>
</tr>
</tbody>
</table>
**Class EdrData**

This class contains all the values that an EDR (alarm and CDR) have.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public String getValue(String key)</td>
<td>Gets the value associated with the specified key.</td>
</tr>
<tr>
<td>public List&lt;String&gt; getValues(String key)</td>
<td>Gets the values associated with the specified key.</td>
</tr>
</tbody>
</table>

**Class ConfigDescriptor**

This class is the parent class of EdrConfigDescriptor, AlarmConfigDescriptor and CdrConfigDescriptor.

**Class EdrConfigDescriptor**

This class contains the data that is specified in the descriptors in the edr.xml configuration file: the identifier and the description.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public long getIdentifier()</td>
<td>Returns the identifier of the EDR.</td>
</tr>
<tr>
<td>public String getDescription()</td>
<td>Returns the description of the EDR.</td>
</tr>
</tbody>
</table>
Creating an EDR listener

**Class AlarmConfigDescriptor**

This class contains the data that is specified in the descriptors in the alarm.xml configuration file: the identifier, the severity and the description.

**Table 31-5 AlarmConfigDescriptor**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public long getIdentifier()</td>
<td>Returns the identifier of the alarm.</td>
</tr>
<tr>
<td>public String getSeverity()</td>
<td>Returns the severity of the alarm.</td>
</tr>
<tr>
<td>public String getDescription()</td>
<td>Returns the description of the alarm.</td>
</tr>
</tbody>
</table>

**Class CdrConfigDescriptor**

This class identifies a CDR. This descriptor does not contain any additional data.

**Updating EDR configuration files**

If using external EDR listeners, and the alarm.xmk, cdr.xml, or edr.xml files have been updated in Network Gatekeeper, the corresponding files needs to be updated in `edrjmslistener.jar`. Replace the existing xml file with the updated file in the edr directory in `edrjmslistener.jar`. 

Network Gatekeeper is deployed in WebLogic Server as two J2EE enterprise applications. One application executes in the access tier and one in the network tier.

As of version 3.0 MP01, you can upgrade Network Gatekeeper in two different ways: a complete shutdown and restart, or a rolling upgrade. The first option will cause the system to be unavailable during the upgrade, while the second option allows traffic to continue to flow.

Performing a rolling upgrade involves the following steps:

1. Prepare the domain.
2. Deploy the new version.
3. Move servers to the new cluster.
4. Cleanup the old application.

### Prepare the domain

To perform a rolling upgrade, you need to have two sets of clusters, that is two access tier clusters and two network tier clusters. The first cluster set holds the old version of Network Gatekeeper, while the second holds the new version. Set up the parallel cluster pairs when Network Gatekeeper is installed. You will only need to do this one time.

**Note:** This step involves a restart of all the servers so it is recommended that you do this directly after an installation of Network Gatekeeper.
Create the clusters that will hold the updated version

Starting in the Administration Console:

1. Select Environment --> Clusters.
2. Click Lock & Edit
3. In Summary of Clusters, select New.
4. Enter the Name of the new cluster and a new, unique Multicast Address and Multicast Port.
5. Click OK.
   Do this for both the network tier cluster and the access tier cluster.
6. In the Change center, click Activate changes.

Edit the config.xml file

1. Shutdown the Administration server and all running servers.

2. Copy the file $DOMAIN_HOME/configuration/jms/wlng-jms.xml to $DOMAIN_HOME/configuration/jms/wlng2-jms.xml

3. On the Administration server, edit the file $DOMAIN_HOME/configuration/config.xml.

4. Configure JMS resources to the new cluster. Add a new JMS resource named WLNGJMSResource_new and target it to the new cluster and the new JMS configuration file for Network Gatekeeper. For example, if the JMS resource is named WLNGJMSResource_new and new network tier cluster is named WLNG_NT_Cluster_new, the following needs to be added to config.xml:

   <jms-system-resource>
   <name>WLNGJMSResource_new</name>
   <target>WLNG_NT_Cluster_new</target>
   <descriptor-file-name>jms/wlng2-jms.xml</descriptor-file-name>
   </jms-system-resource>

5. In the tag <jdbc-system-resource>, for the data source wlng.datasource, add the new clusters as targets. For example, if the new network tier cluster is named
Prepare the domain

WLNG_NT_Cluster_new and the new access tier cluster is named WLNG_AT_Cluster_new, the definition would be:

```xml
<jdbc-system-resource>
  <name>wlng.datasource</name>
  <target>WLNG_NT_Cluster,WLNG_AT_Cluster,WLNG_NT_Cluster_new,WLNG_AT_Cluster_new</target>
  <descriptor-file-name>jdbc/wlng-jdbc.xml</descriptor-file-name>
</jdbc-system-resource>
```

6. In the tag `<jdbc-system-resource>`, for the data source `wlng.tx.datasource`, add the new network tier cluster as a target. For example, if the new network tier cluster is named `WLNG_NT_Cluster_new`, the definition would be:

```xml
<jdbc-system-resource>
  <name>wlng.localTX.datasource</name>
  <target>WLNG_NT_Cluster,WLNG_NT_Cluster_new</target>
  <descriptor-file-name>jdbc/wlng-localTX-jdbc.xml</descriptor-file-name>
</jdbc-system-resource>
```

7. If using Oracle RAC, also do the same updates as described in step 5 and step 6 for the data sources `wlng.datasource-1`, `wlng.datasource-2`, `wlng.localTX.datasource-1`, and `wlng.localTX.datasource-2`.

8.

9. Start the servers.

For more information on creating a WebLogic Cluster, see Setting up WebLogic Clusters.

Define migration settings

Starting in the Administration Console:

1. Select Environment -> Clusters.
2. Click Lock & Edit
3. In Summary of Clusters, click on the new network tier cluster.
4. Click on the Migration tab.
5. In the **Migration Basis** drop-down menu, select Database.

6. In the **Data Source For Automatic Migration** drop-down menu, select wlng.datasource.

7. In the **Auto Migration Table Name** entry field, enter WLS_ACTIVE.

8. In the Change center, click **Activate changes**.

### Deploy the new version

The upgrade requires a new version of both the `wlng_nt.ear` and the `wlng_at.ear` even if only one of them has been updated. The access tier cluster and the new network tier cluster will not find each other unless both are upgraded.

### Modify the file APP-INF/classes/wlng.properties

The file `APP-INF/classes/wlng.properties` exists in both EARs and should be identical. You must change the properties `wlng.at.app_name` and `wlng.nt.app_name` so that they are different than the equivalent ones in the currently deployed EAR files.

**Note:** The names defined in these properties must be used as application names during deployment.

To modify the file `APP-INF/classes/wlng.properties`, explode the ear file using unzip or other tool, make the changes and re-package the ear file, for example using `zip -r <name of ear file> <name of directory>`.

Deploy the new EAR files to the new network tier and access tier clusters. If any WebLogic Server specific configuration was made to the old applications, this should now be done to the new application.

### Move initial access tier and network tier server to the new cluster

Before the updates can be deployed to the new cluster, the first access tier and network tier servers need to be moved to the new clusters. Perform the steps described in “Move the servers to the new cluster” on page 32-5 for these two servers. Then continue with the steps described in Install the updates in the new clusters below.

### Install the updates in the new clusters

Starting in the Administration Console:
Move the servers to the new cluster

1. Select **Deployments**.
2. Click **Lock & Edit**
3. Click **Install** and use the **Install Application Assistant** to select the new ear file to be deployed.
4. In **Choose targeting style** select the target **Install this deployment as an application**.
5. Choose the cluster (network tier cluster and access tier cluster, respectively) as the deployment target.
6. Enter the name of name of the application exactly as defined in `APP-INF/classes/wlmg.properties` in the ear file.
7. Click **Finish**.
   Do this for both the network tier cluster and the access tier cluster.
8. In the Change center, click **Activate changes**.
9. Any other configuration (like WS-Security) that has been done on the Network Gatekeeper must be done manually prior to the upgrade.

For more information on deploying applications, see Install applications and modules.
If you are using the command line deployment tool, see Deploying Applications and Modules with weblogic.deployer.

**Move the servers to the new cluster**

This step involves moving all of the servers from the old cluster to the new. Always move one server from the access tier cluster and one from the network tier cluster at the same time. This will ensure that the clusters are able to find at least one other server that can process any new requests.

Repeat these steps for each server in the domain:

1. Shut down the server process: see Shutdown servers in a cluster.
2. Remove the server from the cluster:
   a. In the Administration Console, select **Environment** and then **Clusters**.
   b. Click **Lock & Edit**
e. Select the old cluster.
d. Select the General tab.
e. Remove the address to the server in the field Cluster Address.
f. Select the Servers tab.
g. Check the server to remove from the cluster and click Remove.

**Note:** Do not remove the server from the domain.

3. Add the server to new cluster:
   a. In the Administration Console, select Environment and then Clusters.
   b. Select the new cluster.
   c. Select the General tab.
   d. Add the address of the server in the field Cluster Address.
   e. Select the Servers tab.
   f. Check the server to add to the cluster and click Add.
   g. Click Activate Changes.

4. Start the server process: see Start Managed Servers from the Administration Console.

5. Start the updated application.

**Cleanup the old applications**

You must undeploy the old EARs. For more information on undeploying an application, see Managing Deployed Applications.

The old, unused clusters can be kept in the configuration for use in any future upgrades.

**Limitations & known issues**

- No configuration changes except the ones related to the upgrade should be performed during the upgrade as this may cause inconsistency in system configuration.
- The old and new network tier cluster will not communicate with each other (except using CORBA). For example, the event channel will not broadcast events.
Limitations & known issues

- After configuration the WebLogic Server Administration Console may indicate that some of the servers need to be restarted for the changes to take affect, but this is not required.
- Some traffic, such as in-flight requests, may be lost during upgrade.
- Requests do not get load balanced between old and new clusters.
- The first network tier and access tier servers that are moved to the new clusters will not be able to process requests until at least one access tier and one network tier server is running.
Rolling Upgrade (MP01)