

# **Oracle® Communication Services Gatekeeper**

System Administrator's Guide

Release 4.0

June 2008

**ORACLE®**

Copyright © 2007, 2008, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

# Contents

## 1. Introduction

Overall Configuration Workflow . . . . .	1-2
--	-----

## 2. Starting and Stopping Servers

Using Scripts . . . . .	2-2
Using Node Manager . . . . .	2-2

## 3. Operation and Maintenance: General

Administration Console Overview . . . . .	3-2
Administration Console . . . . .	3-2
MBeans pane . . . . .	3-3
Configuration and Provisioning . . . . .	3-4
WebLogic Network Gatekeeper Alarms pane . . . . .	3-6
WebLogic Network Gatekeeper CDRs Pane . . . . .	3-9
WebLogic Network Gatekeeper and WebLogic SIP Server management console for SIP-based communication services . . . . .	3-11
Starting the Network Gatekeeper administration console: . . . . .	3-11
Network Gatekeeper Integration Configuration page . . . . .	3-12
Java Management Extensions (JMX) . . . . .	3-13
WebLogic Scripting Tool (WLSLT) . . . . .	3-18
Working in interactive mode. . . . .	3-19
Starting WLST and connecting to Network Gatekeeper . . . . .	3-19
Exiting WLST . . . . .	3-20

Changing an attribute .....	3-20
Invoking an operation .....	3-20
Scripting WLST .....	3-21

## 4. Managing Management Users and Management User Groups

Overview .....	4-1
Users and User Groups .....	4-2
User Types .....	4-4
User Level .....	4-5
Reference: Attributes and Operations for ManagementUsers .....	4-6
Reference: Attributes and Operations for ManagementUserGroup .....	4-9

## 5. Managing and Configuring Budgets

Introduction .....	5-1
Synchronization of budgets .....	5-2
Slave intervals .....	5-3
Master Internal .....	5-3
Failure Conditions .....	5-3
Budget Overrides .....	5-4
Configuration and Management .....	5-4
Reference: Attributes and Operations for BudgetService .....	5-4
Adding a Datasource .....	5-6

## 6. Managing and Configuring EDRs, CDRs and Alarms

About EDRs, CDRs, and Alarms .....	6-1
EDR categories and XML markup .....	6-2
EDR format .....	6-3
EDRs .....	6-5
Alarms .....	6-6

CDRs .....	6-6
External EDR listeners .....	6-6
EDRService .....	6-6
Configuration of the EDRService .....	6-6
Management of the EDRService .....	6-6
Defining batch attributes .....	6-6
Reference: Attributes and Operations for EDRService .....	6-7
Managing EDR, CDR, and alarms configuration files using the EDR Configuration Pane	6-9

## 7. Managing and Configuring Statistics and Transaction Licenses

About Statistics Generation and Reports .....	7-1
Overview of Statistics Reports .....	7-2
System Report to Console .....	7-2
System Report to File .....	7-3
Weekly System Report .....	7-3
License Limit Log Report .....	7-3
Counter Snapshots .....	7-5
Managing StatisticService .....	7-6
Configure the StatisticService .....	7-6
Configure Statistics Types and Transaction Types .....	7-6
View In-Flight Statistics counters .....	7-7
Generate Statistics Reports .....	7-7
Reference: attributes and operations for StatisticsService .....	7-7
Transaction Types .....	7-14

## 8. Setting Up Geographic Redundancy

Introduction .....	8-1
--------------------	-----

Configuration Workflow . . . . .	8-1
Reference: Attributes and Operations for GeoRedundantService. . . . .	8-2

## 9. Managing and Configuring the SNMP service

Introduction . . . . .	9-1
Configuration and management . . . . .	9-3
Configure SNMPService . . . . .	9-3
Trap Receivers . . . . .	9-3
Reference: Attributes and Operations for SNMPService . . . . .	9-3

## 10. Managing and Configuring the Trace Service

Introduction to the Trace Service . . . . .	10-1
Basic tracing. . . . .	10-1
Context trace . . . . .	10-2
Reference: Attributes and Operations for Trace Service . . . . .	10-3
Log4J Hierarchies, Loggers, and Appenders . . . . .	10-9
Configuring Trace for Access Tier servers . . . . .	10-9
Using the Log4J Configuration File . . . . .	10-10
Example Log4J Configuration file. . . . .	10-10

## 11. Managing and Configuring the Storage Service

Introduction to the Storage Service . . . . .	11-1
Reference: Attributes and Operations for Storage Service . . . . .	11-3

## 12. Managing the Policy Service

Introduction . . . . .	12-1
Configuration workflow . . . . .	12-1
Reference: Attributes and Operations for PolicyService . . . . .	12-2

## 13.Setting up WS-Policy and JMX Policy

Introduction . . . . .	13-1
Web Services Security . . . . .	13-1
Configuration workflow: WS-Policy . . . . .	13-2
Apply WS-Policy to a Web Service: Quick start . . . . .	13-3
Setting up UsernameToken with X.509: Quick reference. . . . .	13-4
Setting up UsernameToken with Password Digest: Quick reference . . . . .	13-5
Remove WS-Policy from a Web Service . . . . .	13-8
Create and use custom a custom WS-Policy . . . . .	13-9
Available default WS-Policies. . . . .	13-9
JMX Policy . . . . .	13-10
Administrative Groups . . . . .	13-10
Administrative Service Groups . . . . .	13-10

## 14.Deployment Model for Communication Services and Container Services

Packaging of Communication Services . . . . .	14-2
Version Handling of Communication Services . . . . .	14-5
Deploy and Undeploy Communication Services and plug-ins. . . . .	14-5
Version Handling and Patching of Communication Services. . . . .	14-6
Patch Tool . . . . .	14-7
Examples. . . . .	14-8
Overview of Container Services and Configuration Files. . . . .	14-9
Container services . . . . .	14-10
Retired container services. . . . .	14-11
Disabling of ORB . . . . .	14-12
Patching of Container Services . . . . .	14-13

## 15.Hitless Upgrade Using Production Redeployment

Production Redeployment Overview . . . . .	15-1
Production Redeployment Sequence. . . . .	15-2
Requirements for Production Redeployment . . . . .	15-2
Typical Scenarios for Production Redeployment . . . . .	15-3
Performing a Hitless Upgrade. . . . .	15-5

## 16.Managing and Configuring the Plug-in Manager

Introduction . . . . .	16-1
Execution and evaluation flow. . . . .	16-2
Application-initiated requests. . . . .	16-2
Network triggered Requests . . . . .	16-3
How address ranges are specified when setting up routes. . . . .	16-3
Configuration Workflow. . . . .	16-4
Configuring the Plug-in Manager . . . . .	16-4
Creating a Plug-in instance . . . . .	16-4
Administration of Plug-in Routes and Node IDs . . . . .	16-5
Reference: Attributes and Operations for PluginManager . . . . .	16-5

## 17.Managing and Configuring Third Party Call Communication Services

Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control . . . . .	17-1
Configuration workflow for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control . . . . .	17-2
Management Operations for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control . . . . .	17-3
Properties for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control . . . . .	17-4



Reference: Attributes and Operations for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control. . . . .	17-5
Parlay X 2.1 Third Party Call/INAP . . . . .	17-14
Configuration Workflow for Parlay X 2.1 Third Party Call/INAP . . . . .	17-15
Properties for Parlay X 2.1 Third Party Call/INAP . . . . .	17-16
Management for Parlay X 2.1 Third Party Call/INAP. . . . .	17-17
Reference: Attributes and Operations for Parlay X 2.1 Third Party Call/INAP . .	17-17
Configuration files and dependencies . . . . .	17-21
INAP API configuration file . . . . .	17-22
Common parts configuration file . . . . .	17-25
Back-end configuration file. . . . .	17-26
Parlay X 2.1 Third Party Call/SIP . . . . .	17-27
Configuration Workflow for Parlay X 2.1 Third Party Call/SIP . . . . .	17-28
Management for Parlay X 2.1 Third Party Call/SIP. . . . .	17-29
Properties for Parlay X 2.1 Third Party Call/SIP . . . . .	17-29
Reference: Attributes and Operations for Parlay X 2.1 Third Party Call/SIP . . . .	17-30

## 18. Managing and Configuring Call Notification Communication Services

Parlay X 3.0 Call Notification/Parlay MultiParty Call Control. . . . .	18-1
Configuration Workflow for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control. . . . .	18-2
Properties for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control. . . .	18-4
Management and Provisioning Operations for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control. . . . .	18-5
Reference: Attributes and Operations for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control. . . . .	18-5
Parlay X 2.1 Call Notification/SIP . . . . .	18-9

Properties for Parlay X 2.1 Call Notification/SIP . . . . .	18-10
Configuration Workflow for Parlay X 2.1 Call Notification/SIP . . . . .	18-11
Management for Parlay X 2.1 Call Notification/SIP . . . . .	18-12
Reference: Attributes and Operations Parlay X 2.1 Call Notification/SIP . . . . .	18-12

## 19. Managing and Configuring Short Messaging Communication Services

Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP	19-1
Properties for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP . . . . .	19-2
Configuration Workflow for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP . . . . .	19-4
Management Operations for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP . . . . .	19-6
Reference: Attributes and Operations for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP . . . . .	19-6

## 20. Managing and Configuring Multimedia Messaging Communication Services

Parlay X 2.1 MultiMedia Messaging/MM7 . . . . .	20-1
Properties for Parlay X 2.1 MultiMedia Messaging/MM7 . . . . .	20-2
Configuration Workflow for Parlay X 2.1 MultiMedia Messaging/MM7 . . . . .	20-3
Provisioning Workflow for Parlay X 2.1 MultiMedia Messaging/MM7 . . . . .	20-4
Reference: Attributes and Operations for Parlay X 2.1 MultiMedia Messaging/MM7 . . . . .	20-5

## 21. Managing and Configuring Terminal Location Communication Services

Parlay X 2.1 Terminal Location/MLP . . . . .	21-1
--	------

Properties for Parlay X 2.1 Terminal Location/MLP . . . . .	21-2
Configuration Workflow for Parlay X 2.1 Terminal Location/MLP . . . . .	21-3
Provisioning Workflow for Parlay X 2.1 Terminal Location/MLP . . . . .	21-4
Management Operations for Parlay X 2.1 Terminal Location/MLP . . . . .	21-4
Reference: Attributes and Operations for Parlay X 2.1 Terminal Location/MLP . .	21-4

## 22.Managing and Configuring Audio Call Communication Services

Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction. . .	22-1
Properties for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction . . . . .	22-2
Configuration Workflow for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction. . . . .	22-3
Reference: Attributes and Operations for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction . . . . .	22-5

## 23.Managing and Configuring the Presence Communication Services

Parlay X 2.1 Presence/SIP . . . . .	23-1
URI Cache . . . . .	23-2
Subscriptions Cache . . . . .	23-2
Notifications Cache . . . . .	23-3
Properties for Parlay X 2.1 Presence/SIP . . . . .	23-3
Configuration Workflow for Parlay X 2.1 Presence/SIP . . . . .	23-4
Provisioning Workflow for Parlay X 2.1 Presence/SIP . . . . .	23-5
Management Operations for Parlay X 2.1 Presence/SIP . . . . .	23-5
Reference: Attributes and Operations for Parlay X 2.1 Presence/SIP . . . . .	23-5

## 24.Managing and Configuring Subscriber Profile Communication Services

Extended Web Services Subscriber Profile/LDAPv3 . . . . .	24-1
Properties for Extended Web Services Subscriber Profile/LDAPv3 . . . . .	24-3
LDAP Server Schema . . . . .	24-4
Configuration Workflow for Extended Web Services Subscriber Profile/LDAPv3	24-8
Management for Extended Web Services Subscriber Profile/LDAPv3 . . . . .	24-9
Provisioning for Extended Web Services Subscriber Profile/LDAPv3 . . . . .	24-10
Reference: Attributes and Operations for Extended Web Services Subscriber Profile/LDAPv3 . . . . .	24-10

## 25.Managing and Configuring WAP Push Communication Services

Extended Web Services WAP Push/PAP . . . . .	25-1
Properties for Extended Web Services WAP Push/PAP . . . . .	25-2
Configuration Workflow for Extended Web Services WAP Push/PAP . . . . .	25-3
Management for Extended Web Services WAP Push/PAP . . . . .	25-4
Provisioning for Extended Web Services WAP Push/PAP . . . . .	25-4
Reference: Attributes and Operations for WAP Push/PAP . . . . .	25-4

## 26.Managing and Configuring SOAP2SOAP Communication Services

SOAP2SOAP Communication Services . . . . .	26-1
About SOAP2SOAP Communication Services . . . . .	26-2
Properties for SOAP2SOAP Plug-ins . . . . .	26-3
Configuration Workflow for SOAP2SOAP Plug-ins . . . . .	26-4
Provisioning Workflow for SOAP2SOAP Communication Services . . . . .	26-5
Reference: Attributes and Operations for SOAP2SOAP Plug-ins . . . . .	26-6

## 27. Configuring Heartbeats

Introduction . . . . .	27-1
Configuration and Management . . . . .	27-1
Reference: Attributes and Operations for HeartbeatConfiguration . . . . .	27-2

## 28. Managing and configuring Shortcode mappings

Introduction . . . . .	28-1
Configuration and Management . . . . .	28-2
Management operations . . . . .	28-2
Reference: Attributes and Operations for ShortCodeMapper . . . . .	28-2

## 29. Configuring WebLogic SIP Server Integration

Introduction . . . . .	29-1
Configuration Workflow . . . . .	29-2
Reference: Attributes and operations -General . . . . .	29-3
Reference: Attributes and operations -Third Party Call Control . . . . .	29-4
Reference: Attributes and operations -Call Notification . . . . .	29-4
Attribute: Call Notification Plugin JNDI Name . . . . .	29-5
Attribute: Call Direction Plugin JNDI Name . . . . .	29-5
Reference: Attributes and operations -Presence . . . . .	29-5

## 30. Managing OSA/Parlay Gateway Connections using Parlay\_Access

Understanding OSA/Parlay Gateway and account mappings . . . . .	30-1
Connection model . . . . .	30-1
Information and Certificate Exchange With OSA/Parlay Gateway Administrator . . . . .	30-3
Overall workflow when connecting to an OSA Gateway . . . . .	30-4
Adding an OSA/Parlay Gateway . . . . .	30-5

Adding an OSA Gateway Connection . . . . .	30-5
Creating an OSA client . . . . .	30-6
Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS . . . . .	30-6
Reference: Attributes and Operations for Parlay _Access . . . . .	30-7

# Introduction

This guide describes Network Gatekeeper from the operational, management and provisioning perspectives.

## For Container Services

- [Managing Management Users and Management User Groups](#)
- [Managing and Configuring Budgets](#)
- [Managing and Configuring EDRs, CDRs and Alarms](#)
- [Managing and Configuring Statistics and Transaction Licenses](#)
- [Setting Up Geographic Redundancy](#)
- [Managing and Configuring the SNMP service](#)
- [Managing and Configuring the Trace Service](#)
- [Managing and Configuring the Storage Service](#)
- [Managing the Policy Service](#)
- [Setting up WS-Policy and JMX Policy](#)
- [Deployment Model for Communication Services and Container Services](#)
- [Hitless Upgrade Using Production Redeployment](#)

## For Communication Services

- [Managing and Configuring the Plug-in Manager](#)
- [Managing and Configuring Third Party Call Communication Services](#)
- [Managing and Configuring Call Notification Communication Services](#)
- [Managing and Configuring Short Messaging Communication Services](#)
- [Managing and Configuring Multimedia Messaging Communication Services](#)
- [Managing and Configuring Terminal Location Communication Services](#)
- [Managing and Configuring Audio Call Communication Services](#)
- [Managing and Configuring the Presence Communication Services](#)
- [Managing and Configuring Subscriber Profile Communication Services](#)
- [Managing and Configuring WAP Push Communication Services](#)
- [Configuring Heartbeats](#)
- [Managing and configuring Shortcode mappings](#)
- [Configuring WebLogic SIP Server Integration](#)
- [Managing OSA/Parlay Gateway Connections using Parlay\\_Access](#)

## Overall Configuration Workflow

This section describes an overall workflow for configuring Network Gatekeeper.

1. To setup administrative users: [Managing Management Users and Management User Groups](#).
2. To configure Network Gatekeeper container services:
  - [Managing and Configuring Budgets](#)
  - [Managing and Configuring EDRs, CDRs and Alarms](#)
  - [Managing and Configuring the SNMP service](#)
  - [Managing and Configuring Statistics and Transaction Licenses](#)
  - [Managing the Policy Service](#)



- [Managing and Configuring the Trace Service](#)
  - [Setting up WS-Policy and JMX Policy](#)
  - [Managing and Configuring the Plug-in Manager](#)
  - [Setting Up Geographic Redundancy](#)
3. To configure Network Gatekeeper communication services. **Note:** you only need to configure the ones that are used in your deployment.
- [Managing and Configuring Third Party Call Communication Services](#)
  - [Managing and Configuring Call Notification Communication Services](#)
  - [Managing and Configuring Short Messaging Communication Services](#)
  - [Managing and Configuring Multimedia Messaging Communication Services](#)
  - [Managing and Configuring Terminal Location Communication Services](#)
  - [Managing and Configuring Audio Call Communication Services](#)
  - [Managing and Configuring the Presence Communication Services](#)
  - [Managing and Configuring Subscriber Profile Communication Services](#)
  - [Managing and Configuring WAP Push Communication Services](#)
  - [Managing and Configuring the Plug-in Manager](#)
  - [Managing OSA/Parlay Gateway Connections using Parlay\\_Access](#)
4. To configure Security for Web Services and OAM MBeans:
- [Setting up WS-Policy and JMX Policy](#) for an introduction and pointers to relevant information.
5. To set up geographically redundant site pairs (as necessary):
- [Setting Up Geographic Redundancy](#).

Once the system is configured, move on to the provisioning of service providers and applications, as described in [Creating and maintaining service provider and application accounts in \*Managing Accounts and SLAs\*](#).

## Introduction

# Starting and Stopping Servers

The following sections describe how to start and stop servers in a WebLogic Network Gatekeeper domain:

**Note:** Network Gatekeeper start scripts use default values for many JVM parameters that affect performance. For example, JVM garbage collection and heap size parameters may be omitted, or may use values that are acceptable only for evaluation or development purposes. In a production system, you must rigorously profile your applications with different heap size and garbage collection settings in order to realize adequate performance.

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.

3. Start Access Tier servers in each partition.

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

### Using Scripts

You can start Network and Access Tier servers by using the `startManagedWebLogic` script installed with the Configuration Wizard or a custom startup script.

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
```

**Note:** By default, the servers are started in production mode. This means that user credentials must be provided. There are several ways to do this. See section *Provide User Credentials to Start and Stop Server* in [Starting and Stopping Servers](#) for more informations.

### Using Node Manager

You can also start Network and Access Tier servers by using the Management Console in conjunction with an instance of Node Manager running on each machine. There are many different ways to use Node Manager (see the *Node Manager Administrator's Guide* for more information), but the easiest is to use the Java-based version, as follows:

**Note:** These instructions assume UNIX or Linux. Equivalent Windows versions exist, but Windows is not supported for production servers. These steps must be followed on each of the managed servers.

1. Start the node manager. Best practice is to have this as part of the normal machine boot up sequence. To do it manually, login into the server and change to the `<bea_home>/wlng400/server/bin` directory. Run the `./startNodeManager.sh` script.
2. Edit the `<bea_home>/wlng400/common/nodemanager/nodemanager.domains` file. This file specifies the domains that a Node Manager instance controls. See [Step 4: Configuring nodemanager.domains File](#) for more information. A sample entry:  
`wlng-domain=/bea/user_projects/domains/wlng-domain`
3. Edit the `<bea_home>/wlng400/common/nodemanager/nodemanager.properties` file. Make sure that `StartScriptEnabled=true` is set.
4. Restart the node manager, using Step 1 above.

5. Create a startup script. In the <domain\_home> directory, create a file called startWeblogic.sh and add this line:  

```
./bin/startManagedWebLogic.sh [SERVER_NAME] [ADMIN_HOST_PORT]
```

  
For example: `./bin/startManagedWebLogic.sh WLNG_NT1 192.168.1.42:7001`
6. Make sure that 'Listen Address' is configured in the Console. In **Domain Structure**, click **Environment -> Machines -> <machine\_name>->Node Manager**. You must use **Lock & Edit** to any make any changes.
7. Once everything is set up, to use the Node Manager to start Network Gatekeeper servers, go to the domain's Administration Console. Under **Environment**, select the managed servers you want to start.

There are other numerous ways of starting and stopping servers: for more information see [Starting and Stopping Servers: Quick Reference](#) and [Starting and Stopping Servers](#).

## Starting and Stopping Servers

# Operation and Maintenance: General

The following sections give an overview of the mechanisms available for controlling WebLogic Network Gatekeeper.

- [Administration Console Overview](#)
  - [Administration Console](#)
    - [MBeans pane](#)
    - [Configuration and Provisioning](#)
    - [WebLogic Network Gatekeeper Alarms pane](#)
    - [WebLogic Network Gatekeeper CDRs Pane](#)
  - [WebLogic Network Gatekeeper and WebLogic SIP Server management console for SIP-based communication services](#)
- [Java Management Extensions \(JMX\)](#)
- [WebLogic Scripting Tool \(WLS\)](#)
  - [Working in interactive mode](#)
    - [Starting WLS and connecting to Network Gatekeeper](#)
    - [Exiting WLS](#)
    - [Changing an attribute](#)
  - [Invoking an operation](#)
  - [Scripting WLS](#)

## Administration Console Overview

Network Gatekeeper Administration Console provides a graphical user interface for configuring, managing, and provisioning Network Gatekeeper. The console is an extension to the 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.

## 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.

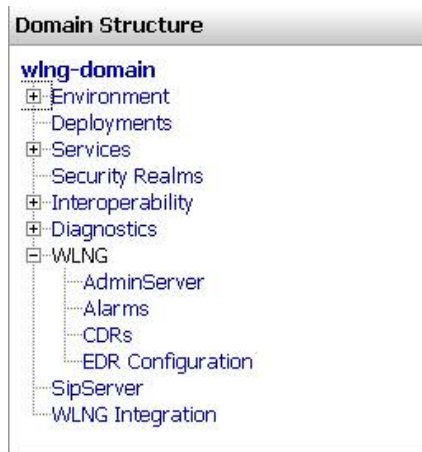
**Note:** If it is the first time Network Gatekeeper is started, use the username `weblogic` and password `weblogic` which is what is recommended when the domain is configured. Once you have logged in, create an administrative user using the instructions given in [Managing Management Users and Management User Groups](#), and remove the user `weblogic`. Either remove or change password for the WebLogic Server user: see <http://edocs.bea.com/wls/docs100/secwres/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 [MBeans pane](#) which displays all OAM objects available for the selected Network Gatekeeper server. Some configuration settings are cluster-wide while other settings are per server.
- Alarms — clicking this link displays the [WebLogic Network Gatekeeper Alarms pane](#).
- CDRs — clicking this link displays the [WebLogic Network Gatekeeper CDRs Pane](#)
- EDR Configuration— clicking this link displays the EDR Configuration pane, see [Managing EDR, CDR, and alarms configuration files using the EDR Configuration Pane](#)



**Figure 3-1 Domain Structure -links to management console for Network Gatekeeper**



## MBeans pane

The MBeans pane contains a tree-structure 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](#).

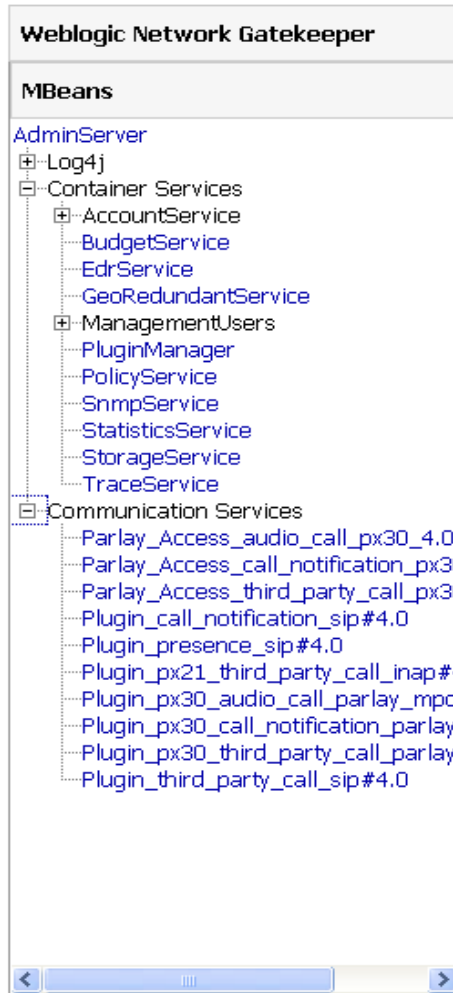
It is possible to expand and collapse this pane by clicking



or



Figure 3-2 MBeans pane



## Configuration and Provisioning

The **Configuration and Provisioning** pane contains two tabs:

- Attributes
- Operations

The **Attributes** tab displays a list of **Attributes**, either read-only or read-write for the managed object.

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

Read-write attributes have a check-box 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 3-3 Example Configuration and provisioning pane -Attributes tab**

The screenshot shows the 'Attributes' tab in the Administration Console. At the top, there are tabs for 'Attributes' and 'Operations'. Below this, the page title is 'Configuration and Provisioning on AdminServer'. The deployment details are: Deployment Name: wlog, Instance Name: PluginManager, and MBean Type: com.bea.wlcp.wlog.plugin.PluginManagerMBean. A note states: 'To make changes to any attributes please select the check box. To invoke an operation, please select the operation from the list.' Below this is a blue 'Update Attributes' button. There are two attribute rows:

Attribute Name	Value	Description
<input type="checkbox"/> PolicyBasedRouting:	false (boolean)	Defines if policy based routing is enabled. This flag determines if node SLAs should be considered in the plug-in selection.
<input type="checkbox"/> ForceConnectInResuming:	false (boolean)	True indicates that the plug-in would not start if it fails to establish communication toward the network in a protocol specific way. Every plug-in in a given .ear must be active, otherwise the .ear would fail to transition to ACTIVE state.  This allows activating a new version of the plug-in only when it is certain that it could accept traffic.

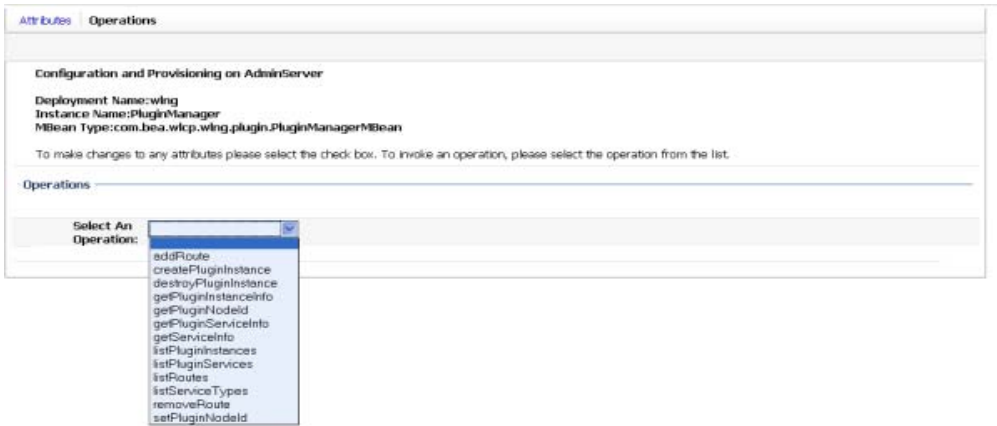
The **Operations** tab contains a drop-down list with the operations for the managed object.

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

To perform an operation:

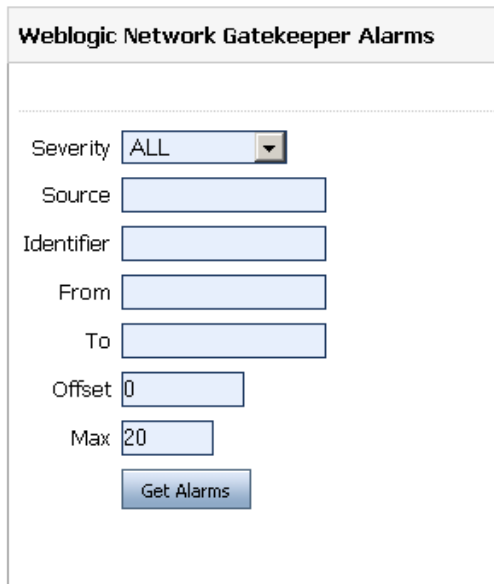
1. Select the operation from the drop-down-list **Select An Operation**.  
The entry-fields for the operation are displayed.
2. Enter the information in the entry fields.
3. Click **Invoke** to perform the operation.

Figure 3-4 Example Configuration and provisioning pane -Operations tab



## WebLogic Network Gatekeeper Alarms pane

Figure 3-5 Alarms pane



The WebLogic Network Gatekeeper Alarms pane displays alarms emitted.

It is possible to filter the output to the page on a set of criteria, see [Table 3-1](#).

The list is returned by clicking **Get Alarms**.

**Table 3-1 WebLogic Network Gatekeeper Alarms pane**

Filter on...	Input
Severity of alarms	From the drop-down list labeled <b>Severity</b> , choose which severity level to display: The options are: <ul style="list-style-type: none"> <li>• <b>ALL</b></li> <li>• <b>WARNING</b></li> <li>• <b>MINOR</b></li> <li>• <b>MAJOR</b></li> <li>• <b>CRITICAL</b></li> </ul>
The server from which the alarm originates	From the drop-down list labeled <b>Server</b> , choose the server from which to display alarms. The options are: <ul style="list-style-type: none"> <li>• <b>ALL</b>, for all servers</li> <li>• <b>&lt;Server name&gt;</b>, for an individual server.</li> </ul>
ID of the alarm.	See <a href="#">Handling Alarms</a> for alarm codes. Use 0 (zero) to wildcard.

**Table 3-1 WebLogic Network Gatekeeper Alarms pane**

Filter on...	Input
Time interval	<p>In the field labeled <b>From</b>, enter the start time for the time interval. The options are:</p> <ul style="list-style-type: none"> <li>• Exact time.</li> <li>• No given start time. Leave empty.</li> </ul> <p>In the field labeled <b>To</b>, enter the end time for the time interval. The options are:</p> <ul style="list-style-type: none"> <li>• Exact time.</li> <li>• No given time. Leave empty.</li> </ul> <p>Exact times are formatted as YYYY-MM-DD hh:mm., where:</p> <ul style="list-style-type: none"> <li>• YYYY is the year. Four digits required.</li> <li>• MM is the month [1...12]</li> <li>• DD is the day [1...[28 29 29 30 31]]. Upper limit depends on month and year.</li> <li>• hh is the hour [0...23]</li> <li>• mm is the minute [0...59]</li> </ul> <p><b>Note:</b> There must be a white space separating YYYY-MM-DD and hh:mm.</p>
Offset	<p>In the field labeled <b>Offset</b>, enter the start offset in the list of alarms entries matching the criteria.</p> <p>Integer. 0 (zero) is the first entry.</p>
Max	<p>In the field labeled <b>Max</b>, enter the maximum number of alarm entries to return.</p> <p>Integer.</p>

## WebLogic Network Gatekeeper CDRs Pane

Figure 3-6 CDRs pane

**Weblogic Network Gatekeeper CDRs**

---

Service Name

Application Id

Service Provider Id

From

To

Offset

Max

```

ApplicationId=app_domain1
ApplicationInstanceGroupId=domain_user
ChargingInfo=null
ConnectTime=0
ContainerTransactionId=BEA1-0016A0C249782835DFE7
DestinationParty=tel:1234
DurationOfUsage=0
EndOfUsage=0
OriginatingParty=tel:7878\\7878\\tel:7878
ServerName=WLNG_NT2
ServiceCorrelationId=null
ServiceName=MultimediaMessaging
ServiceProvider=default_provider
SessionId=null
StartOfUsage=0
TimeStamp=1203444921778
TransactionId=1688849877041399

ApplicationId=app_domain1
ApplicationInstanceGroupId=domain_user
ChargingInfo=null

```

The WebLogic Network Gatekeeper CDRs pane displays CDRs that have been generated.

It is possible to filter the output to the page on a set of criteria, see [Table 3-1](#).

The list is returned by clicking **Get CDRs**.

**Table 3-2 WebLogic Network Gatekeeper CDRs pane**

Filter on...	Input
Service Name	Enter the service name to get CDRs for. The service name is the service type defined for the network protocol plug-in. Leave empty to wildcard.
Application Id	Enter the application ID to filter on. Leave empty to wildcard.
Service provider Id	Enter the service provider ID to filter on. Leave empty to wildcard.
From	Enter the start time for the time interval. The options are: <ul style="list-style-type: none"> <li>• Exact time.</li> <li>• No given start time. Leave empty.</li> </ul> Exact time is formatted as YYYY-MM-DD hh:mm:ss., where: <ul style="list-style-type: none"> <li>• YYYY is the year. Four digits required.</li> <li>• MM is the month [1...12]</li> <li>• DD is the day [1...[28 29 29 30 31]]. Upper limit depends on month and year.</li> <li>• hh is the hour [0...23]</li> <li>• mm is the minute [0...59]</li> <li>• ss is the seconds [0...59]</li> </ul> <p><b>Note:</b> There must be a white space separating YYYY-MM-DD and hh:mm:ss.</p>



**Table 3-2 WebLogic Network Gatekeeper CDRs pane**

Filter on...	Input
To	<p>Enter the end time for the time interval. The options are:</p> <ul style="list-style-type: none"> <li>• Exact time.</li> <li>• No given time. Leave empty.</li> </ul> <p>Exact times are formatted as YYYY-MM-DD hh:mm:ss., where:</p> <ul style="list-style-type: none"> <li>• YYYY is the year. Four digits required.</li> <li>• MM is the month [1...12]</li> <li>• DD is the day [1...[28 29 29 30 31]]. Upper limit depends on month and year.</li> <li>• hh is the hour [0...23]</li> <li>• mm is the minute [0...59]</li> <li>• ss is the seconds [0...59]</li> </ul> <p><b>Note:</b> There must be a white space separating YYYY-MM-DD and hh:mm:ss.</p>
Offset	<p>Enter the start offset in the list of alarms entries matching the criteria.</p> <p>Integer. 0 (zero) is the first entry.</p>
Max	<p>Enter the maximum number of alarm entries to return.</p> <p>Integer.</p>

## WebLogic Network Gatekeeper and WebLogic SIP Server management console for SIP-based communication services

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 communication services, and handles settings for the modules deployed in WebLogic SIP server.

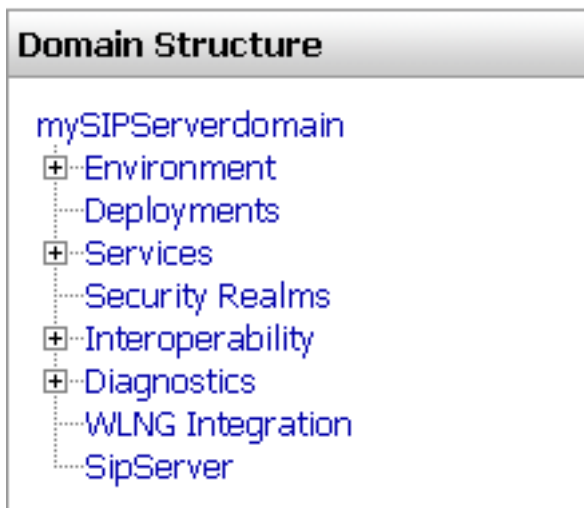
### Starting the Network Gatekeeper administration console:

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 3-7 Domain Structure -links to management console for Network Gatekeeper SIP Server integration modules deployed in WebLogic SIP Server**



## Network Gatekeeper Integration Configuration page

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

These attributes use the Lock and Edit functionality provided by WebLogic Server Administration console.

Figure 3-8 Network Gatekeeper Integration Configuration page in WebLogic SIP server

WLNG Integration Configuration	
<b>General</b>	<a href="#">Third Party Call Control</a>   <a href="#">Call Notification</a>   <a href="#">Presence</a>
<input type="button" value="Save"/>	
WLNG Connection Information	
<b>WLNG JNDI URL:</b>	<input type="text" value="t3://localhost:7002"/>
<b>Username:</b>	<input type="text" value="weblogic"/>
<b>Password:</b>	<input type="text" value="weblogic"/>
<input type="button" value="Save"/>	

## Java Management Extensions (JMX)

Network Gatekeeper exposes its management interfaces as JMX MBeans.

These MBeans come in two flavours:

- Standard MBeans
- Configuration MBeans

**Note:** A link to JavaDoc for OAM can be found in the [Reference](#) topic page in the Network Gatekeeper documentation set on edocs.bea.com. See the reference section for each OAM service for the fully qualified MBean names and how they map to managed objects in Network Gatekeeper Administration Console

Below is an example of connecting to the MBean server and performing an operation on the JMX interfaces.

**Listing 3-1 Example of using JMX to manage Network Gatekeeper**

---

```
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();
```

## Operation and Maintenance: General

```
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 mserverName = args[4]; //NT server name
```

```

String operationName = "addRoute";

String id = "Plugin_px21_multimedia_messaging_mm7";

String addressExpression = ".*";

String[] params = new String[]{id, addressExpression};

String[] signature = new String[]{"java.lang.String", "java.lang.String"};

ObjectName on;

try {

    on = new
ObjectName("com.bea.wlcp.wlmg:Name=wlmg,InstanceName=PluginManager,Type=
com.bea.wlcp.wlmg.plugin.PluginManagerMBean,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();

}

}
```

---

If setting an attribute or calling an operation fails a

`com.bea.wlcp.wlmg.api.management.ManagementException`, or a subclass of this exception is thrown. The following subclasses are defined:

`DuplicateKeyException` - Thrown when the operation results in creating a duplicate key

`InputManagementException` - Thrown for invalid user input

`KeyNotFoundException` - Thrown when the operation cannot find the specified key

A JMX client must have the same version of exception classes available; otherwise the client will throw `ClassNotFoundException`.

`$BEA_Home/wlmg_et400/lib/wlmg/oam.jar` contains custom classes for OAM and return types.

`$BEA_Home/wlmg_et400/doc/javadoc_oam` contains JavDoc for OAM.

## WebLogic Scripting Tool (WSLT)

The following section gives examples of how to use the WebLogic Scripting Tool (WSLT) in both interactive and script mode when configuring and managing Network Gatekeeper.

For information about Weblogic Server and WSLT, see [http://edocs.bea.com/wls/docs100/config\\_scripting/using\\_WSLT.html](http://edocs.bea.com/wls/docs100/config_scripting/using_WSLT.html)

The following topics are covered:



- Working in interactive mode
  - Starting WLSLT and connecting to Network Gatekeeper
  - Exiting WLSLT
  - Changing an attribute
  - Invoking an operation
- WebLogic Scripting Tool (WSLT)

**Note:** A link to JavaDoc for OAM can be found on the [Reference](#) topic page in the Network Gatekeeper documentation set on edocs.bea.com.

## Working in interactive mode

### Starting WLSLT 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 WLSLT: `java weblogic.WLSLT`

3. Connect to the server to manage:

```
connect(' <username>', ' <password>', ' t3://<host>:<port>')
```

4. Change to the custom tree where Network Gatekeeper MBeans are located:

```
custom()
cd('com.bea.wlcp.wlng')
```

5. Display a list of the Mbeans: `ls()`

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.wlmg:Name=wlmg,InstanceName=EdrService,Type=com.bea.wlcp.wlmg.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: `set('<name of attribute>', <value of attribute>)`  
For example, to change the attribute `BatchSize` to 2001 in the managed object `EDRService`: `set('BatchSize', 2001)`
3. 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.
3. Define the data types as an array.
4. Invoke the operation.

For example, to invoke the `displayStatistics` method in the `EDRService` MBean:

```
cd('com.bea.wlcp.wlmg:Name=wlmg,InstanceName=EdrService,Type=com.bea.wlcp.wlmg.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: to add a route using the MBean for the Plug-in Manager:

```
cd('com.bea.wlcp.wlmg:Name=wlmg,InstanceName=PluginManager,Type=com.bea.wlcp.wlmg.plugin.PluginManagerMBean ')
objs=jarray.array(['Plugin_px21_multimedia_messaging_mm7','.*'],Object)
```

```

strs=jarray.array(['java.lang.String', 'java.lang.String'],String)
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, the `invoke` command can be executed as follows:

```

objs =
jarray.array([java.lang.String('stringInput1'), java.lang.String('stringInput2'),
java.lang.String('StringInput3')], java.lang.Object)

strs = jarray.array(['java.lang.String', 'java.lang.String',
'java.lang.String'], java.lang.String)

invoke('methodName', objs, strs)

```

## Scripting WLST

When using WLST together with scripts, WLST is invoked as follows:

```

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 by argument.

### Listing 3-2 Example of script

---

```

userName = sys.argv[1]
passWord = sys.argv[2]
url="t3://" + sys.argv[3] + ":" + sys.argv[4]
objectName = sys.argv[5]
objectName = "com.bea.wlcp.wlmg:Name=wlmg_nt,Type=" + sys.argv[5]
print objectName

```

## Operation and Maintenance: General

```
connect(userName, password, url)
custom()
cd('com.bea.wlcp.wlmg')
cd(objectName)
```

---

**For example: To invoke the script:**

```
java weblogic.WLST script1.py weblogic weblogic localhost 7001
com.bea.wlcp.wlmg:Name=wlmg,InstanceName=PluginManager,Type=com.bea.wlcp.wlmg.
plugin.PluginManagerMBean
```

# Managing Management Users and Management User Groups

The following section describes how to set up and manage administrative users of Network Gatekeeper.

- [Overview](#)
  - [Users and User Groups](#)
  - [User Types](#)
  - [User Level](#)
- [Reference: Attributes and Operations for ManagementUsers](#)
- [Reference: Attributes and Operations for ManagementUserGroup](#)

## Overview

Management of Network Gatekeeper is performed by administrative users. There are a set of management users, identified by their *user type*. Each management user is also assigned a *user level*.

Below is an overview of the operations for managing management users.

To...	Use
Create an administrative user	<a href="#">Operation: addUser</a>
List administrative users	<a href="#">Operation: listUsers</a>
Change password	<a href="#">Operation: changeUserPassword</a>
Delete an administrative user	<a href="#">Operation: deleteUser</a>

## Users and User Groups

Network Gatekeeper classifies its users as either Traffic users or Management users.

- Traffic users are users (Application Instances) who use the application-facing interfaces to send traffic.
- Management users are users who have access to and can perform management and administration functions.

Traffic users cannot login to the management console or perform any management operations.

During installation, the following default groups are created in WebLogic Server Embedded LDAP server

**Table 4-1 User groups**

Group Name	Membership	Role
TrafficUser	All Application Instances belong to this group <ul style="list-style-type: none"> <li>• They should be able to just send traffic and should not have access to management functions.</li> <li>• They should not have access to WebLogic Server or Network Gatekeeper MBeans.</li> <li>• They should not be able to log into the console and perform WebLogic Server administration operations.</li> </ul>	TrafficUser
OamUser	Management users who are of OAM type: <ul style="list-style-type: none"> <li>• They have access to the console based on their level.</li> <li>• They should not be able to send traffic.</li> </ul>	OamUser
PrmUser	Management users who are of type PRM: <ul style="list-style-type: none"> <li>• They should not have access to the console.</li> <li>• They should perform their management operations using the PRM interfaces</li> </ul>	PrmUser

When an Application Instance sends a SOAP request to the application-facing interfaces, it is authenticated by the *WLNG Application Authenticator*; and upon successful authentication it

adds *WLNGTrafficUsers* group to the user principals, in addition to the service provider ID, application ID, service provider group ID, and application group ID.

When management users login successfully, they are added to the *oamUser* group.

Each group contains a user or set of users and is associated with a Security Role. Groups are generally static; they do not change at runtime.

A basic role condition can include users or user groups in a particular security role. For example: *set Admin Role to all users in Administrators group.*

Roles are evaluated at runtime by the Role Mapping Provider by checking the authenticated subject.

A policy contains one or more conditions. For example a simple policy can be: *Allow access if the user belongs to Admin Role.*

## User Types

Below are the pre-defined management user types:

- **Administrative users** use the Administration Console or JMX to interact with Network Gatekeeper.
- **PRM operator users** use the PRM Operator Web Services interfaces to interact with Network Gatekeeper.
- **PRM service provider users** use the PRM Service Provider Web Services interfaces to interact with Network Gatekeeper.

When creating a management user, the user is mapped to the Weblogic Server authentication provider WLNG OAM Authenticator.



## User Level

User level	Access on Network Gatekeeper	Access on WebLogic Server
1000	Administration access to management functions.	Administration access: <ul style="list-style-type: none"> <li>View, modify and administer server configuration</li> <li>Deploy applications</li> <li>Start, resume and stop servers</li> </ul>
666	Read-write access on management functions.	Deployer access: <ul style="list-style-type: none"> <li>View the server configuration, including some encrypted attributes related to deployment activities.</li> <li>Change startup and shutdown classes, Web applications, JDBC data pool connections, EJB, Java EE Connector, Web Service. If applicable, edit deployment descriptors.</li> <li>Access deployment operations in the Java EE Deployment Implementation (JSR-88).</li> </ul>
333	Read-only access on management functions.	Monitor access: <ul style="list-style-type: none"> <li>View the server configuration</li> <li>Have read-only access to Administration Console, WLST and other MBean APIs</li> </ul>
0	No access to management functions. Assigned to PRM Service Provider users internally.	Anonymous access: No access to console

Management users are assigned different user levels based on which JMX resources they will be able to access. At a more granular level, an administrator may want to give access to only a subset of management interfaces. This can be achieved by applying XACML policies.

Below is an outline of how to apply these policies, in order to add more granular access control:

1. Add a new management user.

2. Create a user group.
3. Add the user to the user group
4. Add a XACML policy to assign role to the group
5. Add a XACML policy to the user group. It is possible to restrict access at a granular level; MBean, MBean attribute, or MBean operation level. See <http://edocs.bea.com/wls/docs100/secwires/xacmlusing.html> for a detailed description of this process. The basic process includes:
  - Determine a special identifier, the *resourceId*, for each MBean.
  - Create a XACML policy for the new 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 using the *resourceId*.

## Reference: Attributes and Operations for ManagementUsers

Managed object: Container Services->ManagementUsers->ManagementUsers

MBean: com.bea.wlcp.wlmg.user.management.ManagementUserMBean

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

- [Operation: addUser](#)
- [Operation: changeUserPassword](#)
- [Operation: deleteUser](#)
- [Operation: listUsers](#)

### Operation: addUser

Scope: Cluster

Adds a Network Gatekeeper administrative user.

Signature:

```
addUser(Username:String, Password: String, userLevel: int, type: int)
```

**Table 4-2 addUser**

<b>addUser</b>	
<b>Parameter</b>	<b>Description</b>
Username	User name.
Password	Password
UserLevel	Defines the user level when administrating Network Gatekeeper. See <a href="#">User Level</a> .
Type	Type of management user. Use: 0 for management user 1 for PRM operator user 2 for PRM service provider user See <a href="#">User Types</a> .

## Operation: changeUserPassword

Scope: Cluster

Changes the password for an existing Network Gatekeeper administrative user.

Signature:

```
changeUserPassword(UserName: String, OldPasswd: String, NewPasswd: String)
```

**Table 4-3 changeUserPassword**

<b>changeUserPassword</b>	
<b>Parameter</b>	<b>Description</b>
UserName	User ID for administrative user.
OldPasswd	Current password.
NewPasswd	New password.

## Operation: deleteUser

Scope: Cluster

Deletes a Network Gatekeeper administrative user.

Signature:

```
deleteUser(Username: String)
```

Table 4-4 deleteUser

deleteUser	
Parameter	Description
UserName	User ID for administrative user.

## Operation: getUserLevel

Scope: Cluster

Gets the user level for a management user. See [User Level](#).

Signature:

```
getUserLevel(Username: String)
```

Table 4-5 getUserLevel

getUserLevel	
Parameter	Description
UserName	User ID for the management user.

## Operation: listUsers

Scope: Cluster

Displays a list of all registered management users and their corresponding user levels. See [User Level](#).

```
listUsers(Type: int, Offset: int, Size: int)
```

**Table 4-6 listUsers**

listUsers	
Parameter	Description
Type	Type of user. Use: 0 for Administrative user 1 for PRM Operator user 2 for PRM Service Provider user See <a href="#">User Types</a> .
Offset	Offset in the list. Starts with 0.
Size	Size of the list.

## Reference: Attributes and Operations for ManagementUserGroup

Managed object: Container Services→ManagementUsers→ManagementUserGroup

MBean: com.bea.wlcp.wlmg.user.management.ManagementUserGroupMBean

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

- [Operation: addUserToGroup](#)
- [Operation: createUserGroup](#)
- [Operation: listGroups](#)
- [Operation: listUsers](#)

### Operation: addUserToGroup

Scope: Cluster

## Managing Management Users and Management User Groups

Adds a Network Gatekeeper administrative user to a user group.

Signature:

```
addUserToGroup(Username:String, GroupName: String)
```

**Table 4-7 addUserToGroup**

<b>addUserToGroup</b>	
<b>Parameter</b>	<b>Description</b>
Username	User name.
GroupName	Group name.

## Operation: createUserGroup

Scope: Cluster

Creates a new user group.

Signature:

```
createUserGroup(GroupName: String, Description: String)
```

**Table 4-8 createUserGroup**

<b>createUserGroup</b>	
<b>Parameter</b>	<b>Description</b>
GroupName	Name of the new administrative group.
Description	A textual description.

## Operation: listGroups

Scope: Cluster

Lists all registered user groups.

Signature:

```
listGroups(Offset: int, Size: int)
```

**Table 4-9 listGroups**

<b>listGroups</b>	
<b>Parameter</b>	<b>Description</b>
Offset	Offset in the list. Starts with 0.
Size	Size of the list.

## Operation: listUsers

Scope: Cluster

Lists user based on user group.

Signature:

```
listUsers(GroupName: String, Offset: int, Size: int)
```

**Table 4-10 deleteUser**

<b>deleteUser</b>	
<b>Parameter</b>	<b>Description</b>
GroupName	Group name.
Offset	Offset in the list. Starts with 0.
Size	Size of the list.

## Managing Management Users and Management User Groups



# 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](#)

## 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 periods 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 also 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:

$$r_t = r / (a * n)$$

$$T_t = T / (a * n), \text{ where:}$$

- $r_t$  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.

**T<sub>t</sub>** is the duration between counter synchronization between the slave and the master.

**T** is a time period specified in the SLA.

## 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  $r = 100$ ,  $n = 2$  and  $T = 1000$  milliseconds and  $a = 2$

$r_t = 100 / (2 * 2) = 25$  requests  $T_t = 1000 / (2 * 2) = 250$  milliseconds

The slave synchronizes with the master if the request count reaches 25 or if the time since the last synchronization is 250 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 is not able to account for that server's budget allocation, and can be  $r_t$  requests out-of-sync.

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

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. 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.

Overrides are not enforced across geographically redundant sites.

## Configuration and Management

Configure the following attributes:

- [Attribute: PersistentBudgetFlushInterval](#)
- [Attribute: PersistentBudgetTimeThreshold](#)
- [Attribute: AccuracyFactor](#)
- [Attribute: ConfigUpdateInterval](#)

No management operations are available.

## Reference: Attributes and Operations for BudgetService

Managed object: Container Services→BudgetService

MBean: com.bea.wlcp.wlmg.core.budget.management.configuration.BudgetServiceMBean

Below is a list of attributes for configuration and maintenance.

- [Attribute: PersistentBudgetFlushInterval](#)
- [Attribute: PersistentBudgetTimeThreshold](#)
- [Attribute: AccuracyFactor](#)
- [Attribute: ConfigUpdateInterval](#)

## 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

Under normal operating conditions, Network Gatekeeper, including the Budget service, and the WebLogic Server automatic migration framework share the common transactional (XA) datasource (wlng.datasource) that has been set up for the Network Gatekeeper at large. A datasource is an abstraction that handles connections with the persistent store. Under very heavy traffic it is possible for the Budget singleton service to be deactivated on all servers. This can happen if the 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 singleton 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, see the section *Automatic Migration of User-Defined Singleton Services* in [http://edocs.bea.com/wls/docs100/cluster/service\\_migration.html](http://edocs.bea.com/wls/docs100/cluster/service_migration.html). Also see the section *High-availability Database Leasing* in <http://edocs.bea.com/wls/docs100/cluster/migration.html> 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 the section *Configuring JDBC Data Sources* in [http://edocs.bea.com/wls/docs100/jdbc\\_admin/jdbc\\_datasources.html](http://edocs.bea.com/wls/docs100/jdbc_admin/jdbc_datasources.html).

# 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 and XML markup](#)
  - [EDR format](#)
  - [EDRs](#)
  - [Alarms](#)
  - [CDRs](#)
  - [External EDR listeners](#)
- [EDRService](#)
  - [Configuration of the EDRService](#)
  - [Management of the EDRService](#)
  - [Reference: Attributes and Operations for EDRService](#)
- [Managing EDR, CDR, and alarms configuration files using the EDR Configuration Pane](#)

## About EDRs, CDRs, and Alarms

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

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

## EDR categories and XML markup

EDRs are the base component of both CDRs and alarms: they are, in fact, subsets of EDRs.

In order to categorize the objects in the EDR flow as either pure EDRs, alarms or CDRs, the EDR service uses an EDR configuration file:

```
$DOMAIN_HOME/config/custom/wlng-edr.xml
```

The configuration file contains a set of sections :

- `<edr-config>` contains descriptors that describe pure EDRs.
- `<alarm-config>` contains descriptors that describe EDRs that should be considered alarms.
- `<cdr-config>` contains descriptors that describe EDRs that should be considered CDRs.

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

The XML configuration file can be edited and reloaded using the EDR Configuration pane, see [Managing EDR, CDR, and alarms configuration files using the EDR Configuration Pane](#).

[Listing 6-1](#) illustrates the structure of `wlng-edr.xml`. See [Annotations, EDRs, Alarms, and CDRs](#) in the *Platform Development Studio - Developer's Guide* for more information.

### Listing 6-1 structure of wlng-edr.xml

---

```
<edr-config xsi:schemaLocation="http://www.bea.com/ns/wlng/30
edr-config.xsd">

  <edr id="<ID>" description="<description>">

    <filter>

      <method>

        <name><message response> <message></name>

        <class><fully qualified class name></class>

      </method>
```



```
</filter>  
</edr>  
.....  
</edr-config>
```

---

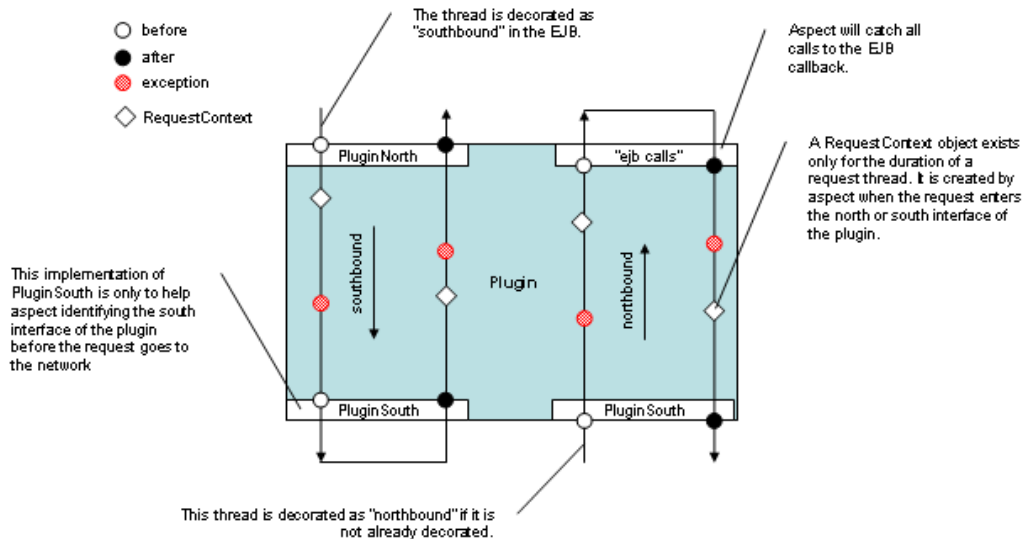
## EDR format

The following values are always available in an EDR when it is 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.

**Figure 6-1 Plug-in north and plug-in south EDR generation**



Below is a description of the contents of an EDR. Individual value fields in an EDR are retrieved by name using a key in a name/value pair.

**Table 6-1 Contents of an EDR**

String value of name (key) in name/ value pair	Description
EdrId	Defined in wlng-edr.xml
ServiceName	The name, or type, of the service.
ServerName	Name of server where the EDR was generated.

**Table 6-1 Contents of an EDR**

<b>String value of name (key) in name/ value pair</b>	<b>Description</b>
Timestamp	The time at which the EDR was triggered. Milliseconds since midnight, January 1, 1970 UTC.
ContainerTransactionId	WebLogic Server transaction ID (if available)
Class	Name of the class that logged the EDR
Method	Name of the method that logged the EDR
Direction	Direction of the request
Source	The type of source that logged the EDR
Position	Position of the EDR relative to the method that logged the EDR
Interface	Interface where the EDR is logged
Exception	Name of the exception that triggered the EDR
SessionId	Session ID
ServiceProviderId	Service provider account ID
ApplicationId	Application account ID
AppInstanceGroupId	Application instance ID.
OrigAddress	The originating address with scheme included. For example tel:1212771234
DestAddress	The destination address, or addresses, with scheme included. May contain multiple addresses.
<custom>	Any additional context-specific information

## EDRs

All EDRs are passed through the EDRService. All EDRs are dispatched to a JMS distributed topic so external clients can receive them over JMS.

EDRs are *not* persisted in the database.

## Alarms

Alarms are EDRs that are mapped to alarms using the alarm.xml configuration file: see [EDR categories and XML markup](#).

Alarms can be configured to be persisted, see [Attribute: StoreAlarms](#).

## CDRs

CDRs are EDRs that are mapped to CDRs using the wlng-edr.xml configuration file, see [EDR categories and XML markup](#).

CDRs can be configured to be persisted, see [Attribute: StoreCDRs](#).

## External EDR listeners

External EDR listeners are JMS topic subscribers: see the documentation for the [Platform Development Studio](#) for information on how to create a EDR listeners.

# EDRService

## 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](#).
2. Specify [Attribute: BatchSize](#).

## Reference: Attributes and Operations for EDRService

Managed object: Container Services→EdrService

MBean: com.bea.wlcp.wlng.edr.management.EdrServiceMBean

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

- [Attribute: BatchTimeout](#)
- [Attribute: StatisticsEnabled](#)
- [Attribute: BatchSize](#)
- [Attribute: PublishToJMS](#)
- [Attribute: StoreAlarms](#)
- [Attribute: StoreCDRs](#)
- [Operation: displayStatistics](#)
- [Operation: resetStatistics](#)

### 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. Must be enabled for [Operation: displayStatistics](#) to be relevant.

### 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. [Attribute: StatisticsEnabled](#) must be true in order for this operation to be relevant.

The following information is displayed:

- Number of EDRs
- Smallest EDR message size in bytes
- Biggest EDR message size in bytes
- Average EDR message size in bytes

Signature:

```
displayStatistics()
```

**Table 6-2 displayStatistics**

displayStatistics	
Parameter	Description
-	-

### Operation: resetStatistics

Scope: Cluster

Resets the statistics for the EDRService.

Signature:

```
resetStatistics()
```

**Table 6-3 resetStatistics**

resetStatistics	
Parameter	Description
-	-

## Managing EDR, CDR, and alarms configuration files using the EDR Configuration Pane

The WebLogic Network Gatekeeper EDR Configuration pane allows the administrator to load new EDR, CDR, and Alarm configuration files.

Open the pane by selecting WLNG-><Server Name>->EDR Configuration from the Domain Structure in the Administration Console.

**Note:** Lock and Edit must be used.

**Table 6-4 WebLogic Network Gatekeeper EDR Configuration pane**

<b>Entry field</b>	<b>Input</b>
EDR descriptor:	The EDR configuration file, see <a href="#">EDR categories and XML markup</a> .
CDR descriptor	The CDR configuration file, see <a href="#">EDR categories and XML markup</a> .
Alarm descriptor	The alarms configuration file, see <a href="#">EDR categories and XML markup</a> .



# 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](#)
  - [System Report to File](#)
  - [Weekly System Report](#)
  - [License Limit Log Report](#)
- [Managing StatisticService](#)
  - [Reference: attributes and operations for StatisticService](#)
- [Transaction Types](#)

## 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 limits enforcement and for grouping reports. Transaction types are in turn grouped into different categories for license enforcement. For more information on transaction types and licensing, see [Licensing](#), a separate document in this set.

Statistics are only generated by communication services. 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.

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 the in-memory store.

These report types are available:

- [System Report to Console](#)
- [System Report to File](#)
- [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.

## System 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 7-1 Structure of license limit log file

---

```
<transaction_limit_log>
  <start> </start>
  <end> </end>
  <log_entry>
```

```
</log_entry>  
<checksum>  
</checksum>  
</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 id 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 7-2 License limit log file example

---

```
<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>

```

---

For more information about licensing, see [Transaction Based Licensing](#).

## Counter Snapshots

Counter snapshots are a real-time snapshot of the statistics counters, to see the counter snapshot for the local server, use [Attribute: CounterSnapshot \(r\)](#)

The snapshot is organized into different categories.

Counter snapshot category	Description
Per server	Sum of the statistics counters for all requests regardless of the originator of the request.
Per server and service provider	Sum of the statistics counters for all requests originating from a given service provider, regardless of application.
Per server and service provider and application	Sum of the statistics counters for all requests originating from a given service provider and application.

The output is in the form of key value pairs, described below.

Key	Value
Source:	Server name.
applicationIdentifier:	Application ID. Empty if counter snapshot category is per server or per server and service provider.

Key	Value
serviceProviderIdentifier:	Service provider ID. Empty if counter snapshot category is per server or per server and service provider.
type:	Statistics type.
num of transactions:	The snapshot of the counter.

## Managing StatisticService

### Configure the StatisticService

To configure...	Use
Statistics persistence interval	<a href="#">Attribute: StoreInterval</a>

### Configure Statistics Types and Transaction Types

Statistics types	Use
Add a new	<a href="#">Operation: addStatisticType</a>
Remove an existing	<a href="#">Operation: removeStatisticType</a>
List existing	<a href="#">Operation: listStatisticTypes</a> <a href="#">Operation: listStatisticTypeDescriptors</a>

## View In-Flight Statistics counters

To view a snapshot of...	Use
a subset of the statistics counters	<a href="#">Attribute: CounterSnapshot (r)</a>

## Generate Statistics Reports

To generate a...	Use
license log report	<a href="#">Operation: createLicenseLimitLog</a>
weekly report	<a href="#">Operation: createWeeklyReport</a>
statistics summary over a time interval	<a href="#">Operation: getStatistics</a>
statistics summary over the last minutes	<a href="#">Operation: getSystemStatistics</a>
statistics report to file	<a href="#">Operation: saveStatisticsToFile</a>

## Reference: attributes and operations for StatisticsService

Managed object: Container Services->StatisticsService

MBean: com.bea.wlcp.wlmg.statistics.management.StatisticsServiceMBean

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

- [Attribute: CounterSnapshot \(r\)](#)
- [Attribute: StoreInterval](#)
- [Operation: addStatisticType](#)
- [Operation: createLicenseLimitLog](#)
- [Operation: createWeeklyReport](#)

- [Operation: getStatistics](#)
- [Operation: getSystemStatistics](#)
- [Operation: listStatisticTypeDescriptors](#)
- [Operation: listStatisticTypes](#)
- [Operation: removeStatisticType](#)
- [Operation: saveStatisticsToFile](#)

**Note:**

**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, by invoking it periodically, can be used to get an approximate value of the traffic throughput. See [Counter Snapshots](#).

**Note:** The counters are reset periodically.

**Attribute: StoreInterval**

Unit: seconds.

Scope: Cluster

Specifies how often statistics data is persisted to the database.

**Operation: addStatisticType**

Scope: Cluster

Adds a statistic type. Used to add a statistics type for custom communication services.

Signature:

```
addStatisticType(Id: int, Name: String)
```



Table 7-1 addStatisticType

addRoute	
Parameter	Description
id	Statistic type ID.
Name	Statistic type name. Descriptive name for the statistic type.

## Operation: createLicenseLimitLog

Scope: Cluster

Creates a license limit supervision log that contains information about the busy hour transaction rate for each transaction group and day during a given time period.

Signature:

```
createLicenseLimitLog(filename: String, startDate: String, endDate: String)
```

Table 7-2 createLicenseLimitLog

createLicenseLimitLog	
Parameter	Description
filename	The filename for the report. The file is created on the local file system of the selected server. Must include an absolute path.  <b>Note:</b> The file must not already exist. If it already exists, the operation 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.

## Operation: createWeeklyReport

Scope: Cluster

Creates a weekly report. See [Weekly System Report](#) for a description of the report.

Signature:

```
createWeeklyReport(SpAccountId: String, StartDate: String, FileName:  
String, Decimals: int)
```

**Table 7-3 createWeeklyReport**

createWeeklyReport	
Parameter	Description
SpAccountId	Service provider ID to filter on.  <b>Note:</b> Leave empty to include all service provider account in the report.
StartDate	Specifies the start date for the report. Format is YYYY-MM-DD.
FileName	The filename for the report. Must include an absolute path. The file is created on the local file system of the selected server.  <b>Note:</b> The file must not already exist. If it already exists, the operation will fail.
Decimals	Number of decimals in the entry for transactions/second in the report.

## Operation: getStatistics

Scope: Cluster or Server

Displays a detailed statistics report for a specific service provider and application. Wildcards and filters apply.

Signature:

```
getStatistics(ServerName: String, StatisticType: String, fromDate: String,  
toDate: String)
```

Table 7-4 `getStatistics`

<code>getStatistics</code>	
Parameter	Description
ServerName	Specifies the name of the server to display statistics from. Leave empty to display statistics generated in all servers.
StatisticType	Specifies the statistics type ID for the statistics type to display. See <a href="#">Operation: listStatisticTypes</a> . <b>Note:</b> Use -1 to display all statistics types.
FromDate	Specifies the start date and time for the interval to display. Format is YYYY-MM-DD hh:mm. <b>Note:</b> Leave empty to display all statistics up to the date and time specified in toDate.
ToDate	Specifies the end date and time for the interval to display. Format is YYYY-MM-DD hh:mm. <b>Note:</b> Leave empty to display all statistics generated from the date and time specified in fromDate up to current date and time.
spAccountId	Service provider account ID to filter on. Leave empty to wildcard.
appAccountId	Application account ID to filter on. Leave empty to wildcard.

### 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:

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

**Table 7-5** `getSystemStatistics`

---

<b>getSystemStatistics</b>	
<b>Parameter</b>	<b>Description</b>
minutes	The number of minutes relative to the current time.

---

### **Operation: listStatisticTypeDescriptors**

Scope: Cluster

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

Signature:

```
listStatisticTypeDescriptors()
```

**Table 7-6** `listStatisticTypeDescriptors`

---

<b>listStatisticTypeDescriptors</b>	
<b>Parameter</b>	<b>Description</b>
-	-

---

### **Operation: listStatisticTypes**

Scope: Cluster

Specifies the statistics types included in the statistics reports.

Signature:

```
listStatisticTypes()
```

Table 7-7 listStatisticTypes

listStatisticTypes	
Parameter	Description
-	-

### Operation: removeStatisticType

Scope: Cluster

Removes a statistics type.

Signature:

```
removeStatisticType(statisticsType: int)
```

Table 7-8 listStatistics

getStatistics	
Parameter	Description
statisticsType	ID for the statistics type.

### Operation: saveStatisticsToFile

Scope: Cluster/Server

Saves a statistics report to file.

Signature:

```
saveStatisticsToFile(serverName: String, StatisticType: String, fromDate: String, toDate: String, filename: String)
```

**Table 7-9 saveStatisticsToFile**

<b>saveStatisticsToFile</b>	
<b>Parameter</b>	<b>Description</b>
serverName	Specifies the name of the server. Leave empty for statistics generated in all servers.
StatisticsType	ID for the statistics type. <b>Note:</b> Use -1 to display all statistics types.
fromDate	Specifies the start date and time for the time interval. Format is YYYY-MM-DD hh:mm. <b>Note:</b> Leave empty for all statistics up to the date and time specified in toDate.
toDate	Specifies the end date and time for the time interval. Format is YYYY-MM-DD hh:mm. <b>Note:</b> Leave empty for all statistics generated from the date and time specified in fromDate up to current date and time.
FileName	Specifies the filename for the report. Must include an absolute path. The file is created on the local file system of the selected server. <b>Note:</b> The file must not already exist. If it does already exist, the method will fail.

## Transaction Types

A set of transaction types are defined for the communication services that come as a part of Network Gatekeeper. The transaction type TRANSACTION\_TYPE\_EXTENSION is used for new communication services, developed as extensions to Network Gatekeeper.

Which events generate which statistics for a certain transaction type are described in the [Communication Service Reference](#), in the sub-section *Statistics* for the communication service in question.

The information includes correlation maps between methods being invoked from either an application or the telecom network and the corresponding transaction type, see the sub-section *Statistics* in the [Communication Service Reference](#).

## Managing and Configuring Statistics and Transaction Licenses



# 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 the [“Redundancy, Load Balancing, and High Availability”](#) chapter in *Concepts and 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 per site, so the remote site must be defined in the local site.

## Configuration Workflow

Below is an outline for configuring the geographically 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 problems in accessing the remote site in [Attribute: RemoteSiteReachabilityAlarmThreshold](#).
3. Define the remote site in [Operation: setSiteAddress](#).

## Reference: Attributes and Operations for GeoRedundantService

Managed object: Container Services→GeoRedundantService

MBean:

com.bea.wlcp.wlmg.core.budget.management.configuration.GeoRedundantServiceMBean

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 its peer site before raising an 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

Signature:

```
getSiteAddress(Site name: String)
```

Displays the address of a given site.

**Table 8-1** getSiteAddress

getSiteAddress	
Parameter	Description
Site name	The name of the site.

## Operation: listRemoteSites

Scope: Cluster

Signature:

```
listRemoteSites()
```

Displays a list of registered remote sites.

**Table 8-2** listRemoteSites

listRemoteSites	
Parameter	Description
-	-

## Operation: removeSite

Scope: Cluster

## Setting Up Geographic Redundancy

Signature:

```
removeSite(Site name: String)
```

Removes a site definition.

**Table 8-3** removeSite

removeSite	
Parameter	Description
Site name	The site name.

## Operation: setSiteAddress

Scope: Cluster

Specifies the address of a remote site.

Signature:

```
setSiteAddress(Site name: String, Address: String)
```

**Table 8-4** setSiteAddress

setSiteAddress	
Parameter	Description
Site name	Name of the remote site.
Address	JNDI URL of the Network Tier for the remote site, according to WebLogic Server addressing standards:  <protocol>://<host>:<port>  Example:  t3://host1:port,host2:port

# Managing and Configuring the SNMP service

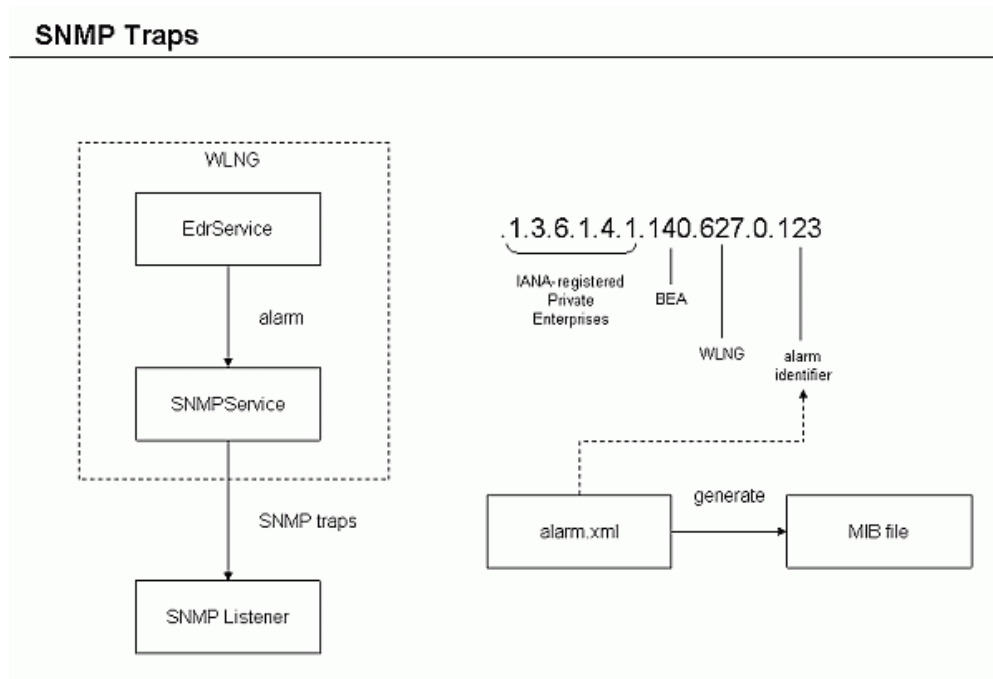
The following section describes how to configure and manage the SNMP service.

- [Configuration and management](#)
- [Reference: Attributes and Operations for SNMPService](#)

## Introduction

The SNMP service is responsible for collecting alarms and distributing them as SNMP traps.

Figure 9-1 SNMP overview



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 consist of:

*IANA-registered private enterprise ID + BEA ID + Network Gatekeeper ID + “0” + 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 `id` of the `<alarm>` element.

The MIB file `BEA-WLNG-MIB` is located in the sub-directory `snmp` in the domain directory.

## Configuration and management

### Configure SNMPService

To define	Use
The SNMP Community string	<a href="#">Attribute: Community</a>
Enterprise Object Identifier. This attribute has a default value and should normally not be changed.	<a href="#">Attribute: EnterpriseObjectIdentifier</a>
SNMP version to use.	<a href="#">Attribute: SNMPVersion</a>
Which alarm severity levels that shall be distributed as SNMP traps.	<a href="#">Attribute: SeverityFilter</a>

Define trap receivers using [Operation: addTrapReceiver](#).

### Trap Receivers

To	Use
Add a trap receiver	<a href="#">Operation: addTrapReceiver</a>
List defined trap receivers	<a href="#">Operation: listTrapReceivers</a>
Delete an existing trap receiver	<a href="#">Operation: listTrapReceivers</a> to list a registered trap receivers. Use the ID as an input parameter in <a href="#">Operation: deleteTrapReceiver</a> .

## Reference: Attributes and Operations for SNMPService

Managed object: Container Services->SNMPService

MBean: com.bea.wlcp.wlmg.snmp.SNMPServiceMBean

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

- [Attribute: Community](#)
- [Attribute: EnterpriseObjectIdentifier](#)
- [Attribute: RepeatedTraps](#)
- [Attribute: SNMPVersion](#)
- [Attribute: SeverityFilter](#)
- [Operation: addTrapReceiver](#)
- [Operation: deleteTrapReceiver](#)
- [Operation: listTrapReceivers](#)

## Attribute: Community

Scope: Server

Format: String

Units: n/a

Specifies the SNMP community address.

Default value is private.

## Attribute: EnterpriseObjectIdentifier

Scope: Server

Format: String

Units: n/a

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

Scope: Cluster

Format: int

Units: n/a

Specifies the number of times each SNMP trap is sent to each configured manager.

## Attribute: SNMPVersion

Scope: Cluster

Format: int

Units: not applicable

Specifies the SNMP version to use.

Enter:

- 0 to use SNMP v1.
- 1 to use SNMP v2.

## Attribute: SeverityFilter

Scope: Cluster

Format: int

Units: not/applicable

Specifies the 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:

Enter:

- 4 to generate SNMP traps for CRITICAL alarms.
- 3 to generate SNMP traps for MAJOR and more severe alarms.
- 2 to generate SNMP trap for MINOR and more severe alarms.
- 1 to generate SNMP traps for WARNING and more severe alarms.

## Operation: addTrapReceiver

Scope: Cluster

Adds a receiver for the SNMP traps. Returns an ID for the receiver.

Signature:

```
addTrapReceiver(Address: String, Port: int)
```

**Table 9-1** addTrapReceiver

addTrapReceiver	
Parameter	Description
Address	IP-address for the trap receiver.
Port	Port for the trap receiver.

## Operation: deleteTrapReceiver

Scope: Cluster

Deletes a previously added trap receiver.

Signature:

```
deleteTrapReceiver(id: int)
```

**Table 9-2** deleteTrapReceiver

deleteTrapReceiver	
Parameter	Description
Id	ID of the trap receiver to delete. Use <a href="#">Operation: listTrapReceivers</a> to list the IDs.

## Operation: listTrapReceivers

Scope: Cluster

Displays a list of all registered trap receivers.

Signature:

```
listTrapReceivers()
```

**Table 9-3 listTrapReceivers**

<b>listTrapReceivers</b>	
<b>Parameter</b>	<b>Description</b>
-	-

## Managing and Configuring the SNMP service

# Managing and Configuring the Trace Service

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

- [Introduction to the Trace Service](#)
- [Reference: Attributes and Operations for Trace Service](#)
- [Log4J Hierarchies, Loggers, and Appenders](#)
- [Configuring Trace for Access Tier servers](#)
- [Using the Log4J Configuration File](#)

## 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 on which it executes. 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](#)

- [Operation: attachAppender](#)
- [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 narrow the generated log messages to requests that match a given 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, applications and so on.

The workflow for defining a context trace file is:

1. In Network Gatekeeper Administration Console choose **Container Services**→**TraceService**
2. Select [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

Managed object: Container Services→TraceService

MBean: com.bea.wlcp.wlmg.log.management.TraceServiceMBean

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

- [Attribute: TracingEnabled](#)
- [Operation: addContextCategory](#)
- [Operation: addContextFilter](#)
- [Operation: attachAppender](#)
- [Operation: createContextTraceFile](#)
- [Operation: createRootContextTraceFile](#)
- [Operation: resetContextFilters](#)
- [Operation: rollOver](#)

### Attribute: TracingEnabled

Scope: Cluster

Unit: n/a

Unit: n/a

Format: Boolean

Specifies whether trace should be enabled or not.

Use:

- **true** to enable tracing
- **false** to disable tracing

### Operation: addContextCategory

Scope: Cluster

Adds a Log4J context category.

Signature:

```
addContextCategory(Identifier: String, Category: String)
```

**Table 10-1 addContextCategory**

<b>addContextCategory</b>	
<b>Parameter</b>	<b>Description</b>
Identifier	ID of a context trace file.
Category	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 <code>com.bea.wlcp.wlmg.plugin.sms.smpp</code>

## 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 10-2 addContextFilter**

<b>addContextFilter</b>	
<b>Parameter</b>	<b>Description</b>
Identifier	ID of a context trace file.
Type	One of the defined types to filter on. Use: <ul style="list-style-type: none"> <li>• SERVICE_PROVIDER to filter on service provider ID.</li> <li>• APPLICATION to filter on application ID.</li> <li>• APPLICATION_INSTANCE to filter on application instance ID.</li> <li>• SESSION to filter on session ID.</li> <li>• TRANSACTION to filter on transaction ID.</li> </ul>
Value	Value to filter on for this filter type.

## Operation: attachAppender

Scope: Cluster

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

Signature:

```
attachAppender(LoggerName: String, AppenderName: String)
```

**Table 10-3 attachAppender**

<b>attachAppender</b>	
<b>Parameter</b>	<b>Description</b>
LoggerName	Name of logger.
AppenderName	Name of Network Gatekeeper service to attach the appender to.

## Operation: createContextTraceFile

Scope: Cluster

Creates a new context trace file.

Signature:

```
createContextTraceFile(Identifier : String, Category: String, Threshold:  
String)
```

**Table 10-4** createContextTraceFile

createContextTraceFile	
Parameter	Description
Identifier	ID of the context trace file to create. Also the name of trace file. The filename has the suffix <code>.log</code> . Must be unique per managed server.
Category	The Log4J category.
Threshold	The threshold for the context trace file. Valid values are: <ul style="list-style-type: none"><li>• OFF</li><li>• FATAL</li><li>• ERROR</li><li>• WARN</li><li>• INFO</li><li>• DEBUG</li><li>• TRACE</li><li>• ALL</li></ul>

## Operation: createRootContextTraceFile

Scope: Cluster

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

Signature:

```
createRootContextTraceFile(Identifier: String)
```

**Table 10-5 createRootContextTraceFile**

<b>createContextTraceFile</b>	
<b>Parameter</b>	<b>Description</b>
Identifier	ID of the context trace file. Also the name of trace file. The filename has the suffix <code>.log</code> . Must be unique per managed server.

## Operation: flushBuffers

Scope: Cluster

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

Signature:

```
flushBuffers()
```

**Table 10-6 flushBuffers**

<b>flushBuffers</b>	
<b>Parameter</b>	<b>Description</b>
-	-

## Operation: removeContextTraceFile

Scope: Cluster

Removes a context trace file.

Signature:

```
removeContextTraceFile(Identifier: String)
```

**Table 10-7** `removeContextTraceFile`

<b>removeContextTraceFile</b>	
<b>Parameter</b>	<b>Description</b>
Identifier	ID of context trace file to remove.

## Operation: `resetContextFilters`

Scope: Cluster

Resets all filters associated with a context trace file.

Signature:

```
resetContextFilters(Identifier: String)
```

**Table 10-8** `resetContextFilters`

<b>resetContextFilters</b>	
<b>Parameter</b>	<b>Description</b>
Identifier	ID of context trace file to reset all filters for.

## Operation: `rollOver`

Scope: Cluster

Rotates log files. This will invoke the `rollOver()` method on all registered `RollingFileAppenders`.

Signature:

```
rollOver()
```

Table 10-9 rollOver

rollOver	
Parameter	Description
-	-

## Log4J Hierarchies, Loggers, and Appenders

There is a set of Log4J Dynamic MBeans shipped with Network Gatekeeper that come by default with appenders. One Log4J hierarchies is defined and displayed in the following entries in the Network Gatekeeper management console under **Log4J**:

- log4j:hiarchy=default

To the default hierarchy, the logger rootDefault is connected, also displayed in the list:

- log4j:Location=AdminServer,logger=rootDefault

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

A set of default appenders are defined:

- log4j:appender=default,Location=AdminServer
- log4j:appender=default,Location=AdminServer

**Note:** The Log4J attributes are dynamic and not persisted.

To persist Log4J settings, use the configuration file:

```
$DOMAIN_HOME/log4j/log4jconfig.xml
```

## 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 [WebLogic Scripting Tool \(WLSLT\)](#) 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:hierarchy=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')`

## Using the Log4J Configuration File

In addition to using the Log4J MBeans, trace can be configured using the file:

```
$DOMAIN_HOME/log4j/log4jconfig.xml
```

The file contains by default an empty skeleton.

The settings define in this file are updated every 30th second.

The ordering of the elements must be taken into account. Appenders must be declared before categories.

When using appenders that writes to file, the files are stored in `$DOMAIN_HOME`.

## Example Log4J Configuration file

Below is an example where the following named appenders are declared:

- `MyRootLogger`
- `MyMultimediaMessagingWarningLogger`

- MyAccountContainerLogger

The all use the same class, `org.apache.log4j.FileAppender`.

Each appender writes to a file which name is given in the `<param name="File" value="<File name>" />` element. The appenders also use the same layout class, `org.apache.log4j.PatternLayout`, and `ConversionPattern`.

The appenders are used by a set of categories, where the name attribute defines which class to trace. The categories also define which Log4J priority that shall be logged.

In the example, the following services are logged:

Everything under:

- `com.bea.wlcp.wlng.account`, that is the Account service.
- `com.bea.wlcp.wlng.plugin.multimediamessaging`, that is the Parlay X Multimedia Messaging communication service.

is logged.

### Listing 10-1 Example log4jconfig.xml

---

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/"
debug="false">
  <appender name="MyRootLogger" class="org.apache.log4j.FileAppender">
    <param name="File" value="A1.log"/>
    <param name="Append" value="false"/>
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%t %-5p %c{2} - %m%n"/>
    </layout>
  </appender>
  <appender name="MyMultimediaMessagingWarningLogger"
class="org.apache.log4j.FileAppender">
    <param name="File" value="MMS_WARN.log"/>
```

## Managing and Configuring the Trace Service

```
<param name="Append" value="false"/>
<layout class="org.apache.log4j.PatternLayout">
  <param name="ConversionPattern" value="%t %-5p %c{2} - %m%n"/>
</layout>
</appender>

<appender name="MyAccountContainerLogger"
class="org.apache.log4j.FileAppender">
  <param name="File" value="Account.log"/>
  <param name="Append" value="false"/>
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d %-5p [%t] %C{2} (%F:%L) -
%m%n"/>
  </layout>
</appender>

<category name="org.apache.log4j.xml">
  <priority value="info"/>
</category>

<!-- log for a certain service... -->
<category name="com.bea.wlcp.wlng.account">
  <priority value="DEBUG"/>
  <appender-ref ref="MyAccountContainerLogger"/>
</category>

<!-- end log for a certain service... -->
<!-- log for a certain service... -->
<category name="com.bea.wlcp.wlng.plugin.multimediamessaging">
  <priority value="WARN"/>
  <appender-ref ref="MyMultimediaMessagingWarningLogger"/>
</category>
```



```
<!-- end log for a certain service... -->  
<root>  
  <priority value="debug"/>  
  <appender-ref ref="MyRootLogger"/>  
</root>  
</log4j:configuration>
```

---

## Managing and Configuring the Trace Service

# Managing and Configuring the Storage Service

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

## Introduction to the Storage Service

The Storage Service provides transparent data access to communication services and container services. Conceptually, a module uses the Storage Service, which uses one or more underlying storage providers that define how and where data is stored. A module creates and uses one or more named stores which can be of different store types. The current storage provider is the Invalidating Storage Provider, an in-memory cache backed by a persistence store, currently a database, acting as the master. The store is a write-through cache. In order to maintain a coherent view of the cache, invalidating events are broadcast within the cluster for specific operations.

Table 11-1 outlines the store types.

**Table 11-1 Store types**

Type	Description
Cluster cache	<p>A cluster wide in memory only store, ideally with redundancy support by keeping backup copies on one or more of the other servers in the cluster. As this is an in memory only store, data reads/writes are very fast.</p>
Write behind database store	<p>A cluster wide in memory cache, ideally with redundancy support by keeping backup copies on one or more of the other server in the cluster, and also backed by a database. Updates for the database table for write behind stores are delayed and asynchronously written in batches to the database.</p> <p>This store has similar performance characteristics as the cluster store for data that is available in the cache, but with better availability.</p>
Write through database store	<p>Similar to write behind data store, this is a cluster wide in memory cache of a database table. Updates for the database table for write through stores are done synchronously as part of the store update operation.</p> <p>Updates to data in the store will be slow compared to cluster store, but read operations will be fast if the data is available in cache. This store offers best reliability.</p>
Database log store	<p>Similar to write behind data store, the log store type data updates to the database table are delayed and asynchronously written in batches to the database.</p> <p>This type of store is meant for additive batch writing of write only data. Since this type of store is not meant for reading, updating or deleting data, it does not need to keep a cache.</p> <p>Since this store is only meant for adding data, read performance will be poor, but optimized for add operations by doing database writes in batches.</p>

**Note:** The only store type used in this release is the Write through database store. Each communication service has it's own store definition in the directory

`$DOMAIN_HOME/config/store_schema`

The store definitions are JAR files with a configuration file and class definitions for the stored data.

## Reference: Attributes and Operations for Storage Service

Managed object: Container Services→StorageService

MBean: `com.bea.wlcp.wlng.storage.management.StorageServiceMBean`

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

- [Attribute: Active \(r\)](#)
- [Operation: getProviderName](#)
- [Operation: getStoreSize](#)
- [Operation: getTableName](#)
- [Operation: getTypeId](#)
- [Operation: listStoreNames](#)

### Attribute: Active (r)

Scope: Cluster

Read-only.

Unit: n/a

Format: Boolean

Indicates whether storage service is active or not:

- **true** if active
- **false** if inactive

### Operation: getProviderName

Scope: Cluster

Gets the storage provider name for a given store.

Signature:

```
getProviderName(StoreName: String)
```

**Table 11-2** `getProviderName`

<code>getProviderName</code>	
Parameter	Description
StoreName	The name of the store as specified in the store configuration file.

## Operation: `getStoreSize`

Scope: Cluster

Gets the size of a given store. The size is number of entries.

Signature:

```
getStoreSize(StoreName: String)
```

**Table 11-3** `getStoreSize`

<code>getStoreSize</code>	
Parameter	Description
StoreName	The name of the store as specified in the store configuration file.

## Operation: `getTableName`

Scope: Cluster

Gets the name of the database table that the store maps to, if any.

Signature:

```
getTableName(StoreName: String)
```

**Table 11-4** `getTable_name`

<code>getTable_name</code>	
Parameter	Description
StoreName	The name of the store as specified in the store configuration file.

## Operation: `getTypeId`

Scope: Cluster

Gets the type ID of the store.

Signature:

```
getTypeId(StoreName: String)
```

**Table 11-5** `getTypeId`

<code>getTypeId</code>	
Parameter	Description
StoreName	The name of the store as specified in the store configuration file.

## Operation: `listStoreNames`

Scope: Cluster

Displays a list configured store names.

Signature:

```
listStoreNames()
```

**Table 11-6 listStoreNames**

<b>listStoreNames</b>	
<b>Parameter</b>	<b>Description</b>
-	-



# Managing the Policy Service

The following section describes how to manage the Policy Service:

- [Introduction](#)
- [Configuration workflow](#)
- [Reference: Attributes and Operations for PolicyService](#)

## Introduction

The Policy Service is responsible for the provisioning of rules into the rules engine. The rules engine can perform some of the policy evaluation of requests flowing through Network Gatekeeper. The Policy Service uses rules written in the policy evaluation language IRL, the Ilog Rules Language. The rules operate on data originating in the requests.

Management of the Policy Service mainly involves loading and deleting rules.

A rule file has an ID, that is given when it is loaded into persistent storage. Rule evaluation is performed if this ID corresponds to the plug-in type associated with the request. If there is no match the rule file with the ID DEFAULT is used. Rules exist on two levels, service provider- and application level.

The rules are enforced by a service interceptor, EvaluateILOGPolicy.

## Configuration workflow

No configuration is necessary.

## Reference: Attributes and Operations for PolicyService

Managed object: Container Services→PolicyService

MBean: com.bea.wlcp.wlmg.policy.PolicyServiceMBean

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: viewApplicationRuleFile](#)
- [Operation: viewServiceProviderRuleFile](#)

### Operation: deleteApplicationRuleFile

Deletes a rule file on the application group level.

Signature:

```
deleteApplicationRuleFile(serviceName: String)
```

Table 12-1 deleteApplicationRuleFile

deleteApplicationRuleFile	
Parameter	Description
serviceName	ID of the rule file, same as the plug-in type.

### Operation: deleteServiceProviderRuleFile

Deletes a rule file on the service provider group level.

Signature:

```
deleteServiceProviderRuleFile(serviceName: String)
```

**Table 12-2 deleteServiceProviderRuleFile**

<b>deleteServiceProviderRuleFile</b>	
<b>Parameter</b>	<b>Description</b>
serviceName	ID of the rule file, same as the plug-in type.

## Operation: listApplicationRuleFiles

Displays the IDs of all application group rule files.

Signature:

```
listApplicationRuleFiles()
```

**Table 12-3 listApplicationRuleFiles**

<b>listApplicationRuleFiles</b>	
<b>Parameter</b>	<b>Description</b>
-	-

## Operation: listServiceProviderRuleFiles

Displays the IDs of all service provider group rule files.

Signature:

```
listServiceProviderRuleFiles()
```

**Table 12-4 listServiceProviderRuleFiles**

<b>listServiceProviderRuleFiles</b>	
<b>Parameter</b>	<b>Description</b>
-	-

## **Operation: loadApplicationRules**

Loads a rule file on the application group level.

Signature:

```
loadApplicationRules(serviceName: String)
```

**Table 12-5 loadApplicationRules**

<b>loadApplicationRules</b>	
<b>Parameter</b>	<b>Description</b>
url	URL to the rule file.
serviceName	ID of the rule file, same as the plug-in type.

## **Operation: loadServiceProviderRules**

Loads a rule file on the service provider group level.

Signature:

```
loadServiceProviderRules(serviceName: String)
```

**Table 12-6 loadServiceProviderRules**

<b>loadServiceProviderRules</b>	
<b>Parameter</b>	<b>Description</b>
url:	URL to the rule file.
serviceName	ID of the rule file, same as the plug-in type.

## Operation: viewApplicationRuleFile

Displays the contents of a rule file on the application group level.

Signature:

```
viewApplicationRuleFile(serviceName: String)
```

**Table 12-7 viewApplicationRuleFile**

<b>viewApplicationRuleFile</b>	
<b>Parameter</b>	<b>Description</b>
serviceName	Name of the service for which the rule file is applied.

## Operation: viewServiceProviderRuleFile

Displays the contents of a rule file on the service provider group level.

Signature:

```
viewServiceProviderRuleFile(serviceName: String)
```

**Table 12-8 viewServiceProviderRuleFile**

---

**viewServiceProviderRuleFile**

---

<b>Parameter</b>	<b>Description</b>
serviceName	Name of the service for which the rule file is applied.

---

# 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.

**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 a WebLogic Server mechanism for Web Services Security -WSSE policies:

- *WebLogic Web Services: Security*  
[http://e-docs.bea.com/wls/docs100/webserv\\_sec/index.html](http://e-docs.bea.com/wls/docs100/webserv_sec/index.html)
- *Understanding WebLogic Security*  
<http://edocs.bea.com/wls/docs100/secintro/concepts.html>
- *Web Services Security specifications*  
[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wss](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss)

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 Administration 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:** By default, Network Gatekeeper is pre-configured to use the WS-Security with UsernameToken. 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://edocs.bea.com/wls/docs100/ConsoleHelp/taskhelp/webservices/ConfigureWSPolicyFile.html> for a full description.

Starting in WebLogic Console:

1. In the **Domain Structure** pane, select **Deployments**, and **Summary of Deployments** tab.
2. Expand the Access Tier part of the communication service, for example **wlng\_at\_audio\_call\_px30**
3. Click on a Web Service on which you wish to apply Web Services security: for example, **com.bea.wlcp.wlng.px30.jws.AudioCallPlayMediaWsImpl**. 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 13-1 Plan.xml snippet to be edited

---

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

---

## Setting up UsernameToken with X.509: Quick reference

This section outlines how to setting up WSSE with X.509.

Starting in WebLogic Console:

1. Configure the credential provider for X.509:
  - a. Selecting **wlng-domain** → **Web Service Security** tab
2. Configure the default identity asserter:
  - a. Select **Security Realms** → **myrealm** → **Providers** → **DefaultIdentityAsserter**
  - b. Select the **Common** tab. Make sure that **x.509** is selected under the **Chosen** list in **Active types**.

- c. Select the **Provider Specific** tab. Make sure **Default User Name Mapper Attribute Type** is set to **CN**.
3. Configure the keystore:
  - a. Select **Environment** → **Servers** → **<AdminServer>**.
  - b. Select **Configuration** → **Keystores**
  - c. Decide which type of keystore to use, and follow the instructions in [http://edocs.bea.com/wls/docs100/secmanage/identity\\_trust.html](http://edocs.bea.com/wls/docs100/secmanage/identity_trust.html)

## Setting up UsernameToken with Password Digest: Quick reference

This section outlines how to apply a WSSE policy of the type UsernameToken with Digest to a Web Service endpoint in Network Gatekeeper.

Starting in the Administrative Console:

1. Configure Default Identity Asserter.
  - a. Select **Security realms** → **my realm** → **provider** → **Default Identity Asserter**
  - b. Add **wsse.PasswordDigest** as the active token type.
  - c. Modify the data source Name (if not **wlmg.datasource**)
2. Configure digest authentication by following the instructions in <http://edocs.bea.com/wls/docs100/ConsoleHelp/taskhelp/webservices/webservicesecurity/UsePasswordDigest.html>.
3. Create a custom ws-policy xml file for the password digest. See example [Listing 13-2](#) for an example.
4. For every Web Service:
  - a. Put the custom ws-policy.xml file in the `WEB-INF/policies` directory of the WAR file for the Web Service. Repackage it and redeploy it.
  - b. Add the custom policy for password digest by following the instructions in [Apply WS-Policy to a Web Service: Quick start](#):

Use the WS-Policy file `WsPolicy:UsernameTokenDigestPolicy.xml`

Edit the deployment plan, `plan.xml`, to indicate inbound only for entry

`WsPolicy_policy: UsernameTokenDigestPolicy.xml` in `plan.xml`

**Listing 13-2 Username Token with digest custom policy example (UsernameTokenDigestPolicy.xml)**

---

```
<?xml version="1.0"?>

<!-- WS-SecurityPolicy -->

<wsp:Policy

    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"

    xmlns:wssp="http://www.bea.com/wls90/security/policy"

    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"

    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"

    xmlns:wls="http://www.bea.com/wls90/security/policy/wsee#part "

    >

    <!-- Identity Assertion -->

    <wssp:Identity>
```

```
<wssp:SupportedTokens>

    <!-- Use UsernameToken for authentication -->

    <wssp:SecurityToken IncludeInMessage="true"

TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-to
ken-profile-1.0#UsernameToken">

        <wssp:UsePassword

Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-p
rofile-1.0#PasswordDigest"/>

    </wssp:SecurityToken>

</wssp:SupportedTokens>

</wssp:Identity>

</wsp:Policy>
```

---

## Remove WS-Policy from a Web Service

Network Gatekeeper 2.2 and earlier used a different mode of authentication than the WS-Security model. 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 `wlmg_at.ear` file after you have made your changes.

To remove the policy files from a Web Service:

1. Explode the EAR file for the Access Tier part of the communication service.
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 13-3](#) and [Listing 13-4](#) for before and after snippets.

### Listing 13-3 `weblogic-webservices-policy.xml` with policy entries

---

```
<?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 13-4 weblogic-webservices-policy.xml with policy entries removed**

---

```
<?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 EAR file.
6. Copy the modified ear file to the domain directory of the Administration Server.
7. If you have previously run Network Gatekeeper, you should now re-deploy the EAR file using the Management Console.

## Create and use custom a custom WS-Policy

Section *Creating and Using a Custom WS-Policy File* in

[http://e-docs.bea.com/wls/docs100/webserv\\_sec/message.html](http://e-docs.bea.com/wls/docs100/webserv_sec/message.html) describes how to create and use a custom WS-Policy file. Also see

<http://e-docs.bea.com/wls/docs100/ConsoleHelp/taskhelp/webservices/ConfigureWSPolicyFile.html>.

### 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/docs100/webserv\\_sec/message.html](http://e-docs.bea.com/wls/docs100/webserv_sec/message.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

Administrative users and groups are set up as described in [Managing Management Users and Management User Groups](#). 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. Network Gatekeeper uses the standard WebLogic Server mechanisms for doing this. See [Securing WebLogic Resources Using Roles and Policies](#), particularly the chapters "Using XACML Documents to Secure WebLogic Resources" and "Reference for XACML on WebLogic Server" for a detailed description of the process. For the basic process you must:

- 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`.

## Setting up WS-Policy and JMX Policy

# Deployment Model for Communication Services and Container Services

Communication services are packaged and deployed in EAR files. The EAR files are grouped, all network protocol plug-ins that share the same group of application-facing interfaces, are packaged into the same EAR file.

Container services are packaged and deployed separately from the communication services, and should not be modified.

- [Packaging of Communication Services](#)
  - [Version Handling of Communication Services](#)
  - [Deploy and Undeploy Communication Services and plug-ins](#)
  - [Version Handling and Patching of Communication Services](#)
- [Overview of Container Services and Configuration Files](#)
  - [Container services](#)
  - [Retired container services](#)
  - [Retired container services](#)
  - [Disabling of ORB](#)
  - [Patching of Container Services](#)

## Packaging of Communication Services

The communication services are grouped so that all communication services that share a common set of application-facing interfaces are grouped together. For example, off-the-shelf, Network Gatekeeper is delivered with two communication services for Parlay X 2.1 Third Party Call:

- Parlay X 2.1 Third Party Call/INAP, where INAP is exposed through Parlay X 2.1 Third Party Call interfaces.
- Parlay X 2.1 Third Party Call/SIP, where SIP is exposed through Parlay X 2.1 Third Party Call interfaces.

Each group of communication services is packaged in two separate EAR files:

`wlng_at_<communication service>.ear`, which is the *service facade*. This part consists only of modules shared among the communication service. The `wlng_at_<communication service>.ear` is deployed in the Access Tier.

`wlng_nt_<communication service>.ear`, which is the *service enabler*. This part contains the network protocol plug-ins for the communication service and common modules for the communication service. The `wlng_nt_<communication service>.ear` is deployed in the Network Tier.

The communication services EAR files are located in the `$BEA_HOME/wlng400/applications` directory.

---

## Example

---

The files holding the communication services that share the Parlay X 2.1 Third Party Call communication service layer consist of the following artifacts:

- `wlng_at_third_party_call_px21.ear`
- `wlng_nt_third_party_call_px21.ear`

`wlng_nt_third_party_call_px21.ear` contains, among other modules:

- `Plugin_px21_third_party_call_inap.jar`, which contains the Parlay X 2.1 Third Party Call/INAP plug-in.
- `Plugin_px21_third_party_call_sip.jar`, which contains the Parlay X 2.1 Third Party Call/SIP plug-in. See note below.

Other modules in this EAR are shared between the two network protocol plug-ins: including the common parts that are tied to the implementation of the communication layer, and any libraries and utilities shared among the plug-ins.

**Note:** `Plugin_px21_third_party_call_sip.jar` is not the only artifact needed in order to achieve end-to-end service communication for communications services connecting to the SIP network. There is also a part that is deployed as a SIP servlet in the SIP Server container. In addition, for plug-ins that use HTTP as the transport protocol and handle network-triggered messages from the network node, there is an HTTP servlet. HTTP servlets are packaged as Web Archives (WARs) and are packaged in the network tier EAR for the communication service. The SIP servlet parts are packaged in a set of files: `wlng-integration-management.jar`, `wlng-integration-console-ext.war`, `wlss-int.ear`, and `wlng-security.jar`.

---

When adding or removing a plug-in to or from `wlng_nt_<communication service>.ear`, the EAR must be exploded and the plug-in specific parts being either added or removed.

[Listing 14-1](#) describes the elements in `wlng_nt_<communication service>.ear`.

### Listing 14-1 Contents of `wlng_nt_<communication service>.ear`

---

```
+---APP-INF/lib
    ...
    <communication service>_callback_client.jar
```

```
    /<utilities 1>.jar
    ...
    /<utilities n>.jar
+---META-INF/MANIFEST.MF
    /application.xml
    /weblogic-application.xml
    /weblogic-extension.xml
+---<plug-in 1>.jar
    ...
+---<plug-in n>.jar
+---<plug-in 1>.war
    ...
+---<plug-in n>.war
+---<communication service>_service.jar
```

---

All plug-ins in the service enabler are packaged as individual jar files in the root of the EAR together with the service EJB, `<communication service>_service.jar`.

If a plug-in connects to the telecom network using HTTP and supports network-triggered requests, there is also a corresponding WAR file that contains the servlet.

`APP-INF/lib` contains any JARs that are shared among the plug-ins in the EAR. This includes the client library for the service callback EJB `<communication service>_callback_client.jar` and one or more utility jar files can be present, depending on the type of communication service.

`META-INF/MANIFEST.MF` is a standard manifest file.

`META-INF/application.xml` is the standard deployment descriptor for Enterprise Application Archives.

`META-INF/weblogic-application.xml` is the BEA WebLogic Server-specific deployment descriptor, see [http://edocs.bea.com/wls/docs100/programming/app\\_xml.html](http://edocs.bea.com/wls/docs100/programming/app_xml.html).

`META-INF/weblogic-extension.xml` is a BEA WebLogic Server-specific deployment descriptor.

[Listing 14-2](#) describes the elements in a plug-in jar file.

---

#### Listing 14-2 Contents of <plug-in X>.jar

---

```

+---com/
+---org/
+---META-INF/MANIFEST.MF
+   instancemap
+   srv_depl.xml

```

---

The jar contains the regular elements in a jar and also an `instancemap`, used among other things to instantiate the plug-in specific implementation of the plug-in interface using the `InstanceFactory`. See [InstanceFactory](#) in *Platform Development Studio - Developer's Guide* for more information.

## Version Handling of Communication Services

Communication services are versioned and can be upgraded using in-production deployment.

The versioning is on EAR level, which means that all network protocol plug-ins for a given collection of application-facing interface are redeployed.

The version is specified in the attribute `Weblogic-Application-Version` in `META-INF/manifest.mf` in `wlmg_at_<communication service>.ear` and `wlmg_nt_<communication service>.ear`, respectively.

For more information, see <http://edocs.bea.com/wls/docs100/programming/versioning.html>.

## Deploy and Undeploy Communication Services and plug-ins

Individual communication service are deployed and undeployed as EARs.

See <http://edocs.bea.com/wls/docs100/deployment/understanding.html> for a description of the different deployment options.

The EAR names for each communication service and the JAR names for the network protocol plug-in are found under the heading “*Properties for <Communication service>*” in the section of *Network Gatekeeper Administration Guide* (this guide) which describes the management of the network protocol plug-in for the communication service in question.

For example, the EAR names for the Third Party Call communication service is found in [Properties for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control](#), [Properties for Parlay X 2.1 Third Party Call/SIP](#), and [Properties for Parlay X 2.1 Third Party Call/INAP](#).

The properties section also describes the JAR file for the plug-in and other artifacts, such as third-party libraries, used by the plug-in.

Below is an example on how to undeploy the Parlay X 2.1 Third Party Call communication service:

```
java webllogic.Deployer -adminurl http://<admin host>:<admin port> -user <admin user> -password <admin password -name wlng_nt_third_party_call_px21 -undeploy -graceful
```

If a plug-in has been removed from the EAR, use the mechanism described in [Performing a Hitless Upgrade](#).

## Version Handling and Patching of Communication Services

To upgrade communication services, patches are applied using a patch tool. The patch tool operates on the EAR files for the communication services. The version of the plug-in is the same as the version of the service enabler it is packaged in, the versioning is on EAR file level.

Below is the typical process for applying patches to communication services.

1. The patch, in the form of a JAR file is provided.
2. The patch is applied to the original EAR file which results in a new, patched, EAR. For example, if the original EAR version is 4.0, the version of the new EAR is 4.0\_1. A sequence number should be suffixed to make each version unique.
3. The new EAR file is deployed and the old version is undeployed, using in-production redeployment, see [Hitless Upgrade Using Production Redeployment](#).
4. If another issue is detected for the same EAR a second patch JAR is provided.
5. The patch is applied to the already patched EAR and creates version 4.0\_2.
6. The new patched EAR is deployed.



Alternatively, both patches can be applied at the same time. This will result in the same version as when applying them separately (version 4.0\_2).

## Patch Tool

Patches are applied using an Ant build file: `$BEA_HOME/wlmg400/server/bin/build.xml`

The correct `CLASSPATH` must be setup in order for the patch build file. Run `setWLSEnv.sh` or `setWLSEnv.cmd` located in `$BEA_HOME/wlmg400/server/bin` to setup the environment.

There are two targets available in the build file:

- `patch` - Applies patches to an EAR file.
- `printpatches` - Displays the currently applied patches.

The syntax is:

```
ant [patch | printpatches] -Dsrc= -Ddest= -Dversion= -Dpatchdir=
-Dpatchfiles=
```

**Table 14-1 Arguments to patch build file**

Target/Argument	Description
<code>patch</code>	This target applies a patch.
<code>printpatches</code>	This target displays the currently applied patches. The information includes: <ul style="list-style-type: none"> <li>• Manifest for the EAR</li> <li>• Manifest-Version</li> <li>• Bundle-Name</li> <li>• Created-By</li> <li>• Ant-Version</li> <li>• Weblogic-Application-Version</li> <li>• Bundle-Vendor</li> <li>• Bundle-Version</li> <li>• For each JAR in the EAR, patched class and ID of the patch is displayed.</li> </ul>
<code>src</code>	The EAR file to apply the patch to.

**Table 14-1 Arguments to patch build file**

Target/Argument	Description
dest	The EAR file that will contain the patch.
version	The new version for the EAR. Must be different from the original version.
patchdir	Directory that holds the patches. Paths are relative or absolute.
patchfiles	The JAR files that contains the patches. Wild cards are supported.

The patch tool detects conflicts if multiple patches try to patch the same file. In this situation an error message is displayed and combination (combo) patch must be used.

## Examples

Below are usage examples.

Example	Command
View patch information for a single EAR file.	<code>ant printpatches -Dsrc=original.ear</code>
View patch information for all EARs in specified directory.	<code>ant printpatches -Dsrc=/path_to_ears</code>
Apply a single patch to an EAR.	<code>ant patch -Dsrc=original.ear -Ddest=patched.ear -Dversion=4.0_1 -Dpatchdir=/patches -Dpatchfiles=CR1234.jar</code>
Apply all patches located in the directory /patches.	<code>ant patch -Dsrc=original.ear -Ddest=patched.ear -Dversion=4.0_2 -Dpatchdir=/patches -Dpatchfiles=*.jar</code>

# Overview of Container Services and Configuration Files

Table 14-2 gives an overview of the Network Gatekeeper configuration files.

**Table 14-2 Network Gatekeeper Configuration files**

File	Contains configuration for...
\$DOMAIN_HOME/config/config.xml	<p>WebLogic Server, see <a href="http://edocs.bea.com/wls/docs100/domain_config/config_files.html">http://edocs.bea.com/wls/docs100/domain_config/config_files.html</a>, with the following additions specific to Network Gatekeeper:</p> <p>...a &lt;custom-resource&gt;, with &lt;name&gt;accesstier&lt;/name&gt;, that specifies the location of at.xml.</p> <p>...a &lt;custom-resource&gt;, with &lt;name&gt;networktier&lt;/name&gt;, that specifies the location of nt.xml.</p> <p>...A set of &lt;app-deployment&gt; specific for the deployed communication services. See the <i>Deployment Artifacts</i> section for the relevant communication service in <a href="#">Communication Service Reference</a>.</p> <p><b>Note:</b> Do not edit this file manually. To manage the life-cycle of communication services, use the Weblogic Server tools, such as the Administration Console's Deployment page or the command line tools described in <a href="http://edocs.bea.com/wls/docs100/deployment/managing.html">http://edocs.bea.com/wls/docs100/deployment/managing.html</a></p>
\$DOMAIN_HOME/config/custom/at.xml	<p>Network Gatekeeper Access Tier container and container services.</p> <p>This is a custom resource.</p> <p><b>Note:</b> Do not edit this file.</p>

**Table 14-2 Network Gatekeeper Configuration files**

File	Contains configuration for...
\$DOMAIN_HOME/config/custom/nt.xml	Network Gatekeeper Network Tier container and container services. This is a custom resource. <b>Note:</b> Do not edit this file.
\$DOMAIN_HOME/config/custom/wlng-edr.xml	EDRs, CDRs, and alarms. This is a custom resource <b>Note:</b> Do not edit this file manually.

## Container services

Table 14-3 describes the Network Gatekeeper container services. The interfaces to these, together with common classes, are packaged in

`$BEA_HOME/wlng400/server/lib/wlng/wlng.jar`

The container services are packaged in separate JARs, located in  
`$BEA_HOME/wlng400/modules`

**Table 14-3 Container services**

Container service	Summary
<code>com.bea.wlcp.wlng.core.CoreServerService</code>	Performs initial setup during the start-up sequence.
<code>com.bea.wlcp.wlng.snmp.SNMPServerService</code>	SNMP service.
<code>com.bea.wlcp.wlng.corba.ob.OrbacusServerService</code>	Initializes the Orbacus ORB and makes that the default ORB.
<code>com.bea.wlcp.wlng.event_channel.EventChannelServerService</code>	Broadcasts arbitrary events between modules and servers in a cluster.
<code>com.bea.wlcp.wlng.storage.StorageServerService</code>	Storage service.

**Table 14-3 Container services**

Container service	Summary
com.bea.wlcp.wlng.storage.db.DbServerService	Storage provider for persistence using direct database access.
com.bea.wlcp.wlng.storage.inval.InvalidatingServerService	Storage provider for persistence using an invalidating cache.
com.bea.wlcp.wlng.storage.configuration.ConfigurationStoreServerService	Storage provider for configuration settings.
com.bea.wlcp.wlng.edr.EdrServerService	EDR service.
com.bea.wlcp.wlng.core.budget.BudgetServerService	Budget service.
com.bea.wlcp.wlng.core.budget.GeoRedundantServerService	Geographical redundancy service.
com.bea.wlcp.wlng.account.AccountServerService	Account service.
com.bea.wlcp.wlng.statistics.StatisticsServerService	Statistics service.
com.bea.wlcp.wlng.policy.PolicyServerService	Generic Policy evaluation service. Wraps implementations of rules engines.
com.bea.wlcp.wlng.plugin.PluginManagerServerService	Plug-in manager.

## Retired container services

[Table 14-4](#) describes the retired Network Gatekeeper container services. These services are deprecated and are not deployed as a part of standard Network Gatekeeper installation. The services are packaged in:

- `wlng_at_bc.ear`, see [Table 14-4](#).
- `wlng_nt_bc.ear`, see [Table 14-5](#).

**Table 14-4 Retired services, kept for backwards compatibility in \$BEA\_HOME/wlmg400/applications/wlmg\_at\_bc.ear**

Container service	Summary
Access Web Service	Used by applications to establish sessions. Uses the Access service, see <a href="#">Table 14-5s</a> .

**Table 14-5 Retired services, kept for backwards compatibility in \$BEA\_HOME/wlmg400/applications/wlmg\_nt\_bc.ear**

Container service	Summary
Access_service	Used in connection with the Access Web service exposed to applications.
Slee_alarm	Alarm service used by external CORBA alarm listeners of the type used when integrating with Network Gatekeeper 2.2 and earlier.
Slee_charging	Charging service used by external CORBA charging listeners of the type used when integrating with Network Gatekeeper 2.2 and earlier.
Slee_edr	EDR service used by external CORBA EDR listeners of the type used when integrating with Network Gatekeeper 2.2 and earlier.

## Disabling of ORB

By default, the Orbacus ORB is started when Network Gatekeeper starts.

The ORB binds to the same IP as WebLogic Server binds to. This can have impact on, among other things, the IOR callbacks having `localhost` in them if WebLogic Server is configured to listen to `localhost`.

If a deployment does not use any of the CORBA features, the ORB can be disabled.

When the ORB is disabled none of the EARs that contain modules that require CORBA can be deployed. This includes the EARs that contains any of the [Retired container services](#) and also any communication service EAR that includes an OSA/Parlay-type network protocol plug-in. See the administration section for each communication service for information about which parts of the communication service EAR are related to the OSA/Parlay-type network protocol plug-ins.

To disable the ORB, do the following steps on *all* servers:

1. Shutdown server.
2. Edit `$Domain_home/config/custom/nt.xml`
3. Remove the line:  

```
<service><service-class>com.bea.wlcp.wlmg.corba.ob.OrbacusServerService</service-class></service>
```
4. Start server.

## Patching of Container Services

Patching of container services are done using SmartUpdate, see <http://e-docs.bea.com/common/smartupdate/guide>.

Patches to container services and Weblogic Server is done using rolling upgrades, see [http://edocs.bea.com/common/docs100/upgrade/rolling\\_upgrade.html](http://edocs.bea.com/common/docs100/upgrade/rolling_upgrade.html).

The following patches should not be removed using SmartUpdate Tool. The Patch IDs are:

- DRS1
- EEZI
- WGBA

## Deployment Model for Communication Services and Container Services



# Hitless Upgrade Using Production Redeployment

Network Gatekeeper supports seamless upgrades of communication services without service interruption. The upgrades are performed without disruptions in the traffic flow, without any need to restart servers, which means that the full processing capacity of the Network Gatekeeper cluster is preserved during the hitless upgrade.

Hitless upgrades are performed using the Production Redeployment strategy, also referred to in Weblogic Server as side-by-side redeployment. See <http://edocs.bea.com/wls/docs100/deployment/redeploy.html>

Upgrades of container services provided by Network Gatekeeper do not support production redeployment. Rolling upgrades should be used instead. See [http://edocs.bea.com/common/docs100/upgrade/rolling\\_upgrade.html](http://edocs.bea.com/common/docs100/upgrade/rolling_upgrade.html).

## Production Redeployment Overview

Production redeployment involves deploying a new version of an application alongside an older version of the same application. Weblogic Server automatically manages client connections so that only new client requests are directed to the new version. Clients already connected to the application during the redeployment continue to use the older version of the application until it has processed any pending request, at which point WebLogic Server automatically retires the older application. In this context, applications are network protocol plug-ins.

To support the production redeployment strategy, a plug-in must *graceful shutdown*. This means that it tracks any in-flight requests, and makes sure that it does not get undeployed while having any pending unprocessed requests. The WebLogic Server container takes care of parts of this, but

all plug-ins need to perform protocol specific clean-up of any connections used for network traffic, and assert that all traffic is diverted to a *new version* of the module.

Production redeployments are performed on EAR file level, which means that all network protocol plug-ins that are tied to an application-facing interface are updated.

## Production Redeployment Sequence

Below is a description of the Production Redeployment sequence:

1. The redeployment operation is triggered. It is performed on the administration server and targeted to all servers in the cluster.
2. The old version of the plug-in stops accepting new requests at the same time as the new version of the plug-in starts accepting them. At this point, both plug-ins are operational, the old version does not accept new requests. It only processes pending requests.
3. When the old version has no pending requests, it is undeployed automatically.

While both the new and the old version of version of the plug-in are active, both plug-ins are registered in the plug-in manager, with the version of the plug-in indicated. See [Managing and Configuring the Plug-in Manager](#).

## Requirements for Production Redeployment

For production redeployment to work:

- The old version of the plug-in must:
  - Perform protocol-specific clean-up of any connections or sessions towards the network nodes
  - Assert that all traffic is diverted to the new version of the module.
  - Track pending requests, if the network protocol used by the plug-in is not based on HTTP or RMI. Pending requests over HTTP and RMI are tracked by WebLogic Server mechanisms.

**WARNING:** If this is not done properly, the old version could be undeployed prematurely.

- Report back to the container when ready to be undeployed.
- The new version of the plug-in must not:

- Change the schema used to define the data model for the named store in the StorageService
- Change the interface used between the Access Tier and the Network Tier in the communication service

## Typical Scenarios for Production Redeployment

- Communication services using HTTP-based network protocol plug-ins:

HTTP based network protocol plug-ins do not establish HTTP sessions when sending or receiving requests to the underlying network nodes. In the absence of sessions the Weblogic Server servlet container simply chooses the most recent/active version of a particular WAR module whenever new network triggered requests arrive. In-progress network triggered request are allowed to complete through the shutdown of EAR-scoped work managers. Before the older version of the EAR is retired, all of the pending work related to handling of the network triggered traffic is completed.

- Parlay X 2.1 Short Messaging-Binary SMS/SMPP communication service:

The Parlay X 2.1 Short Messaging-Binary SMS/SMPP plug-in handles graceful transition to ADMIN state as part of the server life-cycle or during EAR upgrade. Draining both application-initiated and network triggered traffic out of the older version is handled internally by the plug-in. Once a new version of the plug-in becomes available, the older version begins the process of completing the processing of all application initiated traffic it is responsible for. When all of the application-initiated traffic has switched to the newer version of the plug-in, the suspension of the network-triggered traffic in the older version is initiated. During the suspension of network triggered traffic, all new requests from the SMSC are rejected with `ESME_RSYSERR` error code in the responses. The in-progress network-triggered requests are acknowledged to the SMSC as they are processed by the application or are stored for later retrieval/processing. For requests that have received the `ESME_RSYSERR` response, the SMSC is expected to fail-over to the newer plug-in version. Application-initiated traffic is handled first in order to minimize the window during which the `ESME_RSYSERR` command status error codes are sent to the SMSC.

The new version of the plug-in will bind new receiver connections before the old version disconnects its receiver connections. This will ensure continuous network triggered traffic. No state is associated with the bind itself and any plug-in on any server is able to handle any network triggered traffic regardless of what server created the notifications. Only the volatile conversational state needs to be handled before the plug-in unbinds any of the ongoing connections toward the SMSC.

Hitless upgrade of both Transmitter/Receiver and Transceiver bind types are supported.

- Communication services using OSA/Parlay type plug-ins

The OSA access functionality is embedded in the Network Tier communication service EAR file, and any upgrades or updates to this part will follow the new EAR.

All application initiated traffic is switched to the newer version of a plug-in as soon as it becomes available. This prevents newer call sessions from being started against the older version. For application initiated traffic, the new OSA/Parlay plug-in might get traffic that concerns sessions started with the previous version in the case of long-lived sessions, such as call control. In this case, the old version must still be able to handle all call-backs from the OSA/Parlay Gateway that might be related to the traffic handled by the new version.

The OSA/Parlay plug-in counts the number of unfinished active sessions handled when it is being retired, and uses a barrier to inform the WebLogic Server container when all sessions have finished. This is the case for the Parlay X 2.1 Third Party Call/MPCC, Call Notification/MPCC, and Audio Call/MPCC-CUI plug-ins, where there are long lived sessions.

Each plug-in also has a time-out to provide a reasonable boundary for the overall graceful retirement duration.

The OSA/Parlay plug-ins use the managers replicated by the OSA Access module to create new notifications and set new call-backs.

- Communication services using SIP type plug-ins

SIP based Communication Services consists of three Gatekeeper plug-ins and one SIP application. Third Party Call, Call Notification, and Presences are bundled in one EAR, wlss-int.ear.

The Gatekeeper side plug-ins can be hitlessly upgraded the same way as other Gatekeeper plug-ins.

Graceful retirement is not supported for SIP applications. For the SIP Servlet part of the plug-ins, Network Gatekeeper supports the same `-retiretimeout` option as WebLogic SIP Server.

- Extended Web Services Subscriber Profile/LDAP communication service

The Subscriber Profile/LDAP plug-in uses synchronous application-initiated traffic only. In-flight traffic is tracked and use a barrier is used to notify the WebLogic Server container when all traffic is completed.

The EAR names for each communication service and the JAR names for the specific network protocol plug-in(s) contained within it can be found under the heading “Properties for <Communication service>” in the section of *Network Gatekeeper Administration Guide* (this

guide). See [Deployment Model for Communication Services and Container Services](#) for a description of how individual communication services are packaged.

## Performing a Hitless Upgrade

Network Gatekeeper uses the WebLogic Server mechanism for production redeployment. See <http://edocs.bea.com/wls/docs100/deployment/redeploy.html> for more information.

Below is the syntax for using `weblogic.Deployer` to do the upgrade:

```
java weblogic.Deployer -adminurl <adminhost>:<adminport> -user <admin user>
-password <password> -graceful -rmiGracePeriod <grace period> -redeploy -name
<name> -source <EAR file> -targets <Cluster name>
```

In addition to the `-graceful` and `-rmiGracePeriod` flags, the `-retiretimeout <time period>` flag can be used to force stop the application after the given time period.

In the code snippet below the Network Tier part of the Parlay X 2.1 Multimedia Messaging communication service on a cluster named `WLNG_NT_Cluster` is redeployed:

```
java weblogic.Deployer -adminurl <adminhost>:7001 -user weblogic -password
weblogic -redeploy -rmiGracePeriod 30 -name wlng_nt_multimedia_messaging_px21
-source ./wlng_nt_multimedia_messaging_px21.ear -targets WLNG_NT_Cluster
```

The EARs are version-controlled, so the `meta-inf\MANIFEST.MF` file in the EAR must be updated with which version the EAR has:

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.6.5
Created-By: R27.5.0-101-94136-1.5.0_14-20080116-2103-windows-ia32 (BEA
Systems, Inc.)
Bundle-Name: wlng_at_multimedia_messaging_px21
Bundle-Version: $<version>
Bundle-Vendor: BEA Systems, Inc
Weblogic-Application-Version: $<version>
```

## Hitless Upgrade Using Production Redeployment

# Managing and Configuring the Plug-in Manager

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 when setting up routes](#)
- [Configuration Workflow](#)

## Introduction

Network protocol plug-ins consist of two parts: the plug-in service and the plug-in instance. The plug-in service is the deployable and updatable unit. It does not itself process any traffic. Plug-in instances are instantiated from a plug-in service. One plug-in service can be the base for many plug-in instances.

A plug-in service:

- Has a unique ID.
- Has a version.
- Is packaged in an EAR, together with other plug-ins of that share the same set of application-facing interfaces.

A plug-in instance:

- Is instantiated from a plug-in service
- Has a unique ID
- Is versioned based on the plug-in service it is instantiated from.
- Belongs to a type.
- Exposes a set of traffic interfaces to the service communication layer. This set is a Java representation of the Web Services interfaces exposed by the service access layer.
- Interfaces with network nodes using one or more protocols.
- Supports a set of address schemes.
- Is configured and managed independently of other instances.
- Can be assigned a node ID to be used when setting up node SLAs.

Plug-in instances are created using the Plug-in Manager MBean.

The plug-in manager introspects the plug-in north interfaces, methods, and names of any arguments. This is useful when creating up service provider group and application group SLAs.

## Execution and evaluation flow

### Application-initiated requests

When an application's request is processed, it is routed to the network protocol plug-in instance via the plug-in manager. The plug-in manager triggers a chain of service interceptors, where the request is evaluated and a given plug-in instance is selected based on:

- Data in the request originating from the application.
- Status of the plug-in (only plug-ins in status active are considered)
- Configured routes
- Properties of the registered plug-ins, including:
  - Plug-in type.
  - Plug-in ID.
  - Address plan, that is the kind of network to which they are connected: for example E.164 or SIP.



- Usage policies based on SLA settings and policy rules.
- Load balancing
- Data from external sources: any custom data-lookups implemented as service interceptors.

Once the decision has been made, the request is forwarded to the selected plug-in.

Routes are configured in the Plug-in manager via OAM operations, and are used by the service interceptors.

Refer to the section in this guide for each communication service for the plug-in service ID and plug-in type under which individual plug-ins are registered.

The following interceptors are use data configured using the plug-in manager:

- RemoveInvalidAddressPlugin
- CreatePluginList

## Network triggered Requests

When a request is triggered from the network, the plug-in manager is also part of the request flow, and is responsible for invoking the chain of service interceptors.

## How address ranges are specified when setting up routes

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 “\.”.

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 zero or more times. 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).

The address scheme can be included in the expression. If not specified, any scheme will match.  
Examples

- tel:^46.\* matches all tel (E.164) numbers starting with 46.
- sip:.\* matches any SIP address.

## Configuration Workflow

Configuring the plug-in manager can be divided into:

- 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 communication services, see [Configuring the Plug-in Manager](#).

## Configuring the Plug-in Manager

Below is an outline for configuring the plug-in manager:

1. Specify whether or not to use policy based routing in [Attribute: PolicyBasedRouting](#). Policy based routing is necessary in order to enforce node SLAs.
2. Specify the network protocol plug-in behavior when it is being deployed or re-deployed in [Attribute: ForceConnectInResuming](#)

## Creating a Plug-in instance

Below is an outline for creating an instance of a plug-in service:

1. Find the plug-in service ID for the service you wish to use to create the plug-in instance. The plug-in service IDs are listed under the heading “Properties” in the sections describing the management of each plug-in. To get a list of plug-in service IDs, use [Operation: listPluginServices](#).

2. Use [Operation: createPluginInstance](#) to create the instance. The Plug-in Instance ID supplied as a parameter will be used to identify the instance for all routing and administration.
3. To destroy an instance use [Operation: destroyPluginInstance](#)

## Administration of Plug-in Routes and Node IDs

The administration of routes uses the following operations:

- To add a new route or change a route: [Operation: addRoute](#)
- To remove a route: [Operation: removeRoute](#)
- To list existing routes: [Operation: listRoutes](#)
- To get information about a specific plug-in instance, [Operation: getPluginServiceInfo](#).
- To get a list of all registered plug-in instances: [Operation: listPluginInstances](#)
- To get a list of all registered plug-in services: [Operation: listPluginServices](#)
- To define the node ID: [Operation: setPluginNodeId](#).

A node ID is assigned to one or more plug-in instance IDs to enforce node SLAs,. The node ID is then referred to in the node SLAs.

## Reference: Attributes and Operations for PluginManager

Managed object: Container Services->PluginManager

MBean: com.bea.wlcp.wlng.plugin.PluginManagerMBean

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

- [Attribute: ForceConnectInResuming](#)
- [Attribute: PolicyBasedRouting](#)
- [Operation: addRoute](#)
- [Operation: createPluginInstance](#)
- [Operation: destroyPluginInstance](#)
- [Operation: getPluginInstanceInfo](#)
- [Operation: getPluginNodeId](#)

- [Operation: getPluginServiceInfo](#)
- [Operation: listPluginInstances](#)
- [Operation: listPluginServices](#)
- [Operation: listRoutes](#)
- [Operation: listServiceTypes](#)
- [Operation: removeRoute](#)
- [Operation: setPluginNodeId](#)

## Attribute: ForceConnectInResuming

Scope: Cluster

Unit: n/a

Format: Boolean

Specifies if a network protocol plug-in service is allowed to transition to state **ACTIVE** if it fails to establish a connection with the underlying network node during deployment or re-deployment. This assures that a new version of the network protocol plug-in service is activated only when it is certain that it can accept traffic.

If enabled, all plug-ins services packaged in the EAR that is being updated must become active before any traffic is routed to the plug-in instances derived from the new version. If any plug-in instance fails to connect to the underlying network, the new version does not become active.

Use:

- **true** to check if a connection can be established before accepting traffic.
- **false** to not perform this check.

## Attribute: PolicyBasedRouting

Scope: Cluster

Unit: n/a

Format: Boolean

Specifies whether policy based routing should be used.

Use:

- **true** to enable policy based routing
- **false** to disable policy based routing

**Note:** Policy based routing must be enabled in order to enforce node SLAs.

## Operation: addRoute

Scope: Cluster

Adds a new plug-in route. A route is identified by the plug-in instance ID and a match pattern.

Signature:

```
addRoute(PluginInstanceId: String, AddressExpression: String)
```

**Table 16-1** addRoute

addRoute	
Parameter	Description
PluginInstanceId	ID of the plug-in instance that is the target of the route. A list of plug-in instance IDs is displayed using the <a href="#">Operation: listPluginInstances</a> .
AddressExpression	The pattern to be used as a matching criteria. See <a href="#">How address ranges are specified when setting up routes</a> .

## Operation: createPluginInstance

Scope: Cluster

Creates a new instance of a specific plug-in service.

Signature:

```
createPluginInstance(PluginServiceId: String, PluginInstanceId: String)
```

**Table 16-2 createPluginInstance**

<b>createPluginInstance</b>	
<b>Parameter</b>	<b>Description</b>
PluginServiceId	ID of the plug-in service to instantiate. See <a href="#">Operation: listPluginServices</a> for a list of plug-in service IDs.
PluginInstanceId	ID of the plug-in instance to be created. The ID must be unique among all instances. All existing instances can be displayed using <a href="#">Operation: listPluginInstances</a> .

## Operation: destroyPluginInstance

Scope: Cluster

Destroys an instance of a specific plug-in instance. All routes associated with the instance are kept.

Signature:

```
destroyPluginInstance(PluginServiceId: String, PluginInstanceId: String)
```

**Table 16-3 destroyPluginInstance**

<b>destroyPluginInstance</b>	
<b>Parameter</b>	<b>Description</b>
PluginServiceId	ID of the plug-in service which has a plug-in instance that is to be destroyed.
PluginInstanceId	ID of the plug-in instance to destroy.

## Operation: getPluginInstanceInfo

Scope: Cluster

Gets information about a specific plug-in instance. The information includes:

- The node ID, if the plug-in instance is assigned a node ID using [Operation: setPluginNodeId](#).
- A list of North interfaces registered with the plug-in manager.
- A list of South Interfaces registered with the plug-in manager.
- Whether the plug-in instance is connected to the network node (true if connected).
- A list of all configured routes

Signature:

```
getPluginInstanceInfo(PluginInstanceId: String)
```

**Table 16-4** getPluginInstanceInfo

<b>getPluginInstanceInfo</b>	
<b>Parameter</b>	<b>Description</b>
PluginInstanceId	ID of the plug-in instance to get information about.

## Operation: getPluginNodeId

Scope: Cluster

Displays which plug-in node ID a plug-in instance is associated with, if any.

Signature:

```
getPluginNodeId(PluginInstanceId: String)
```

**Table 16-5** getPluginNodeId

<b>getPluginNodeId</b>	
<b>Parameter</b>	<b>Description</b>
PluginInstanceId	ID of the plug-in.

## Operation: `getPluginServiceInfo`

Scope: Cluster

Displays information about a plug-in service. The information includes:

- The service type
- A list of address schemes the plug-in service supports
- The protocol it uses towards the network node.
- A list of plug-in instances crated based on the plug-in service.

Signature:

```
getPluginServiceInfo(PluginServiceId: String)
```

**Table 16-6** `getPluginServiceInfo`

<code>getPluginServiceInfo</code>	
Parameter	Description
<code>PluginServiceId</code>	ID of the plug-in service.

## Operation: `getServiceInfo`

Scope: Cluster

Displays information about a service type. The information includes:

- A list of interface names for the service type
- A list of method names for each interface
- A list of argument names for each method.
- The name of the return type for each method.

This list is useful when setting up SLAs.

Signature:

```
getServiceInfo(Type: String)
```



**Table 16-7** getServiceInfo

getServiceInfo	
Parameter	Description
Type	ID of the service type. See <a href="#">Operation: listServiceTypes</a> .

## Operation: listPluginInstances

Scope: Cluster

Displays a list of plug-in instances IDs.

The list is rendered as:

```
<Plug-in instance ID>#<JEE application name>#<version of JEE application>
```

The plug-in instance ID is the part of each entry up to the first #.

Signature:

```
listPluginInstances()
```

**Table 16-8** listPluginInstances

listPluginInstances	
Parameter	Description
-	-

## Operation: listPluginServices

Scope: Cluster

Displays a list of plug-in service IDs. A plug-in service ID uniquely identifies a plug-in service.

Signature:

```
listPluginServices()
```

**Table 16-9 listPluginInstances**

listPluginInstances	
Parameter	Description
-	-

## Operation: listRoutes

Scope: Cluster

Displays a list of all registered routes.

Signature:

```
listRoutes()
```

**Table 16-10 listRoutes**

listRoutes	
Parameter	Description
-	-

## Operation: listServiceTypes

Scope: Cluster

Displays a list of service types. The service types defines a collection of plug-in services exposing a specific functionality in the network.

For example: SMS or MMS.

Signature:

```
listServiceTypes()
```

**Table 16-11 listServiceTypes**

listServiceTypes	
Parameter	Description
-	-

## Operation: removeRoute

Scope: Cluster

Removes a route. The route is identified by the ID of the plug-in instance and the matching pattern.

Signature:

```
removeRoute(PluginInstanceId: String, AddressExpression: String)
```

**Table 16-12 removeRoute**

removeRoute	
Parameter	Description
PluginInstanceId	ID of the plug-in.
AddressExpression	The pattern used as a matching criteria. See <a href="#">How address ranges are specified when setting up routes</a> .

## Operation: setPluginNodeId

Scope: Cluster

Assigns a node ID to a plug-in instance.

Signature:

```
setPluginNodeId(PluginInstanceId :String, nodeId: String)
```

**Table 16-13 setPluginNodeId**

<b>setPluginNodeId</b>	
<b>Parameter</b>	<b>Description</b>
PluginInstanceId	The plug-in instance ID.
nodeId	ID to be used as a node ID. This ID is used when enforcing node SLAs.

# Managing and Configuring Third Party Call Communication Services

The following section describes configuration and maintenance attributes and operations for the communication services that expose Parlay X 3.0 and Parlay X 2.1 Third Party Call Web Services interfaces. The sections also provide a workflow for the configuration. There are three plug-in services related to Third Party Call:

- [Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control](#)
- [Parlay X 2.1 Third Party Call/INAP](#)
- [Parlay X 2.1 Third Party Call/SIP](#)

## Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control

This section contains a description of the configuration attributes and operations available for the Parlay X 3.0 Third Party Call/OSA MultiParty Call Control network protocol plug-in instance.

To see a	Refer to
Detailed list of necessary information for managing and configuring the plug-in instance	<a href="#">Properties for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control</a>
Configuration workflow	<a href="#">Configuration workflow for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control</a>
List of operations and attributes related to management and provisioning	<a href="#">Management Operations for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control</a>
Reference of management attributes and operations	<a href="#">Reference: Attributes and Operations for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control</a>

## Configuration workflow for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control

1. Using either the Management Console or an MBean browser, select the MBean detailed in [Properties for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control](#).
2. Configure the attributes of the plug-in instance:
  - [Attribute: CallingParticipantNameMandantory](#)
  - [Attribute: MaximumDurationEnforced](#)
  - [Attribute: MultiMediaSupported](#)
  - [Attribute: ChargingAllowed](#)
  - [Attribute: StatusRetentionTime](#)
  - [Attribute: ChangeMediaAllowed](#)
  - [Attribute: MaximumParticipants](#)
  - [Attribute: MaximumParticipants](#)
  - [Operation: configCallGetInfoReq](#)

- [Operation: configLegGetInfoReq](#)
3. Gather information about the OSA Gateway and configure the plug-in instance 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.
  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](#).
  5. Setup the routing rules to the plug-in instance, see [Configuring the Plug-in Manager](#). Use the plug-in instance ID and address schemes detailed in [Properties for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control](#).
  6. If desired, create and load a node SLA. See:
    - [Defining Global Node and Service Provider Group Node SLAs](#)
    - [Managing Application SLAs](#)

Move on to provisioning of service provider accounts and application accounts.

## Management Operations for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control

The following operations are related to management:

- [Operation: getCallLegs](#)
- [Operation: getCallSessionInfo](#)
- [Operation: getCallLegSessionInfo](#)

- [Operation: listCallSessionIds](#)
- [Operation: countPendingCallSession](#)

## Properties for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control

Property	Description
Managed object in Administration Console	<domain name>->WLNG-><server name>->Communication Services->Plugin_px30_third_party_call_parlay_mpcc
MBean	Domain=com.bea.wlcp.wlmg Name=wlmg_nt InstanceName=Plugin_px30_third_party_call_parlay_mpcc Type=com.bea.wlcp.wlmg.plugin.tpc.parlay.management.ThirdPartyCallMBean
Network protocol plug-in service ID	Plugin_px30_third_party_call_parlay_mpcc
Network protocol plug-in instance ID	Plugin_px30_third_party_call_parlay_mpcc
Supported Address Scheme	tel
North interface	com.bea.wlcp.wlmg.px30.plugin.ThirdPartyCallPlugin
Service type	ThirdPartyCall
Exposes to the service communication layer a Java representation of:	Parlay X 3.0 Part 2: Third Party Call
Interfaces with the network nodes using:	Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control SCF; Subpart 7: MultiParty Call Control Service
Deployment artifacts	Plugin_px30_third_party_call_parlay_mpcc.jar, packaged in wlmg_nt_third_party_call_px30.ear px30_third_party_call.war, packaged in wlmg_at_third_party_call_px30.ear



This plug-in service does not support multiple instantiation using the Plug-in Manager. There is a one to one mapping between the plug-in service and the plug-in instance. The plug-in instance is created when the plug-in service is started.

## Reference: Attributes and Operations for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control

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

- Attribute: [CallGetInfoReqConfig \(r\)](#)
- Attribute: [CallingParticipantNameMandatory](#)
- Attribute: [LegGetInfoReqConfig \(r\)](#)
- Attribute: [MaximumDurationEnforced](#)
- Attribute: [MultiMediaSupported](#)
- Attribute: [ChargingAllowed](#)
- Attribute: [StatusRetentionTime](#)
- Attribute: [ChangeMediaAllowed](#)
- Attribute: [MaximumParticipants](#)
- Operation: [configCallGetInfoReq](#)
- Operation: [configLegGetInfoReq](#)
- Operation: [countPendingCallSession](#)
- Operation: [getCallLegs](#)
- Operation: [getCallSessionInfo](#)
- Operation: [getCallLegSessionInfo](#)
- Operation: [listCallSessionIds](#)

### Attribute: [CallGetInfoReqConfig \(r\)](#)

Read-only.

Scope: Cluster

Unit: n/a

Format: String

Indicates current configuration for the method `getInfoReq` operation in `IpMultiPartyCall` interface. The information includes:

- **Supported:** Boolean that indicates if the operation is supported or not.
- **PCallInfoTimes:** Boolean that indicates if `P_CALL_INFO_TIMES` tag is present or not in the operation.
- **PCallInfoReleaseCause:** Boolean that indicates if `P_CALL_INFO_RELEASE_CAUSE` tag is present in the operation.

Use [Operation: configCallGetInfoReq](#) to change these settings.

## Attribute: CallingParticipantNameMandantory

Scope: Cluster

Unit: n/a

Format: Boolean

Specifies if the parameter `callingParticipantName` in the method `makeCallSession` defined in the Parlay X 3.0 Third Party Call interface shall be used as the original address or not.

If `callingParticipantName` is required, it must be in the form of a string which can be translated to a URI, for example `tel:123456`.

Enter:

- **True** if `callingParticipantName` shall be used as the originating address
- **False** if not

## Attribute: LegGetInfoReqConfig (r)

Read-only

Scope: Cluster

Unit: n/a

Format: String

Indicates current configuration for the method `getInfoReq` operation in `IpCallLeg` interface. The information includes:

- **Supported:** Boolean that indicates if the operation is supported or not
- **PcallLegInfoTimes:** Boolean that indicates if `P_CALL_LEG_INFO_TIMES` tag is present or not in the operation.
- **PcallLegInfoReleaseCause:** Boolean that indicates if `P_CALL_LEG_INFO_RELEASE_CAUSE` tag is present or not in the operation.
- **PcallLegInfoAddress:** Boolean that indicates if `P_CALL_LEG_INFO_ADDRESS` tag is present or not in the operation.
- **PcallLegInfoAppInfo:** Boolean that indicates if `P_CALL_LEG_INFO_APPINFO` tag is present or not in the operation.

Use [Operation: configLegGetInfoReq](#) to change these settings.

## Attribute: MaximumDurationEnforced

Scope: Cluster

Unit: n/a

Format: Boolean

Specifies if a call which duration exceeds the maximum value shall be terminated or not

Enter:

- **True** to terminate the call.
- **False** to allow the call to continue.

## Attribute: MultiMediaSupported

Scope: Cluster

Unit: n/a

Format: Boolean

Indicates if multimedia is supported

- `true` if multimedia is supported
- `False` if not

## Attribute: ChargingAllowed

Scope: Server

Unit: n/a

Format: Boolean

Specifies whether charging is allowed.

- `true` if an application is allowed to specify charging information when creating a call session (operation `makeCallSession`).
- `False` if not.

## Attribute: StatusRetentionTime

Scope: Server

Unit: Seconds

Format: int [0-]

Specifies the length of time information about a call is stored after the call is terminated.

## Attribute: ChangeMediaAllowed

Scope: Server

Unit: n/a

Format: int Boolean

Specifies if an end user, that is, a call participant, is allowed to change the media used in the call.

- `true` if an end user is allowed to change media for an existing call session. In order for a end user to change the media type for a given call session, this attribute must be true and the application must have allowed it when the call session was established (operation `makeCallSession`) and [Attribute: MultiMediaSupported](#) must be True.
- `False` if not.

## Attribute: MaximumParticipants

Scope: Server

Unit: n/a

Format: int [2-]

Specifies the maximum number of participants in a call.

## Operation: configCallGetInfoReq

Scope: Cluster

Configures the parameters in the getInfoReq operation in the interface IpMultiPartyCall.

Signature:

```
configCallGetInfoReq(Supported: Boolean, PCallInfoTimes: Boolean,
PCallInfoReleaseCause: Boolean)
```

**Table 17-1 configCallGetInfoReq**

<b>configCallGetInfoReq</b>	
<b>Parameter</b>	<b>Description</b>
Supported	Specifies if the operation is supported or not by the OSA/Parlay Gateway.
PCallInfoTimes	Specifies if the P_CALL_INFO_TIMES tag shall be present in TpCallInfoType or not. Use: <ul style="list-style-type: none"> <li>• <b>True</b> to add it to TpCallInfoType</li> <li>• <b>False</b> to not</li> </ul>
PCallInfoReleaseCause	Specifies if the P_CALL_INFO_RELEASE_CAUSE tag shall be present in TpCallInfoType or not. Use: <ul style="list-style-type: none"> <li>• <b>True</b> to add it to TpCallInfoType</li> <li>• <b>False</b> to not</li> </ul>

## Operation: configLegGetInfoReq

Scope: Cluster

Configures the parameters in the getInfoReq operation in the interface IpCallLeg.

Signature:

```
configLegGetInfoReq(Supported : Boolean, PCallLegInfoTimes : Boolean,
PCallLegInfoReleaseCause : Boolean, PCallLegInfoAppInfo : Boolean)
```

**Table 17-2 configLegGetInfoReq**

<b>configLegGetInfoReq</b>	
<b>Parameter</b>	<b>Description</b>
Supported	Specifies if the operation is supported or not by the OSA/Parlay Gateway.
PCallLegInfoTimes	Specifies if the P_CALL_LEG_INFO_TIMES tag shall be present in TpCallLegInfoType or not. Use: <ul style="list-style-type: none"> <li>• <b>True</b> to add it to TpCallLegInfoType</li> <li>• <b>False</b> to not</li> </ul>
PCallLegInfoReleaseCause	Specifies if the P_CALL_LEG_INFO_RELEASE_CAUSE tag shall be present in TpCallLegInfoType or not. Use: <ul style="list-style-type: none"> <li>• <b>True</b> to add it to TpCallLegInfoType</li> <li>• <b>False</b> to not</li> </ul>
PCallLegInfoAddress	Specifies if the P_CALL_LEG_INFO_ADDRESS tag shall be present in TpCallLegInfoType or not. Use: <ul style="list-style-type: none"> <li>• <b>True</b> to add it to TpCallLegInfoType</li> <li>• <b>False</b> to not</li> </ul>
PCallLegInfoAppInfo	Specifies if the P_CALL_LEG_INFO_APPINFO tag shall be present in TpCallLegInfoType or not. Use: <ul style="list-style-type: none"> <li>• <b>True</b> to add it to TpCallLegInfoType</li> <li>• <b>False</b> to not</li> </ul>

## Operation: getCallLegs

Scope: Cluster

Displays a list of IDs for all call legs in a call session.

Signature:

```
getCallLegs(CallSessionId: String)
```

**Table 17-3 getCallLegs**

getCallLegs	
Parameter	Description
CallSessionId	ID of the call session to list call legs for.

## Operation: getCallSessionInfo

Scope: Cluster

Displays information about a call session. This includes:

- `callSessionId`: the ID of the call session.
- `callStatus`: the current status of the call. The status is one of :
  - `Established`, during call duration.
  - `Terminated`, when the call has terminated but information is still present in the internal storage, see [Attribute: StatusRetentionTime](#).
  - `Expired`, when the call is terminated and information is no longer available.
- `originalAddress`: the originator (a-party) of the call.
- `appInstGrpId`: the application instance ID associated with the application that created the call session.
- `callRef`: the CORBA reference to the call object in the Parlay Gateway (`IpMultiPartyCall`).
- `srcPlugin`: the type of plug-in instance that initiated the call. The type is one of:
  - `ThirdPartyCall`

- CallNotification
- CallDirection

Signature:

```
getCallSessionInfo(CallSessionId: String)
```

**Table 17-4 getCallSessionInfo**

<b>getCallSessionInfo</b>	
<b>Parameter</b>	<b>Description</b>
CallSessionId	ID of the call session to get information about.

## Operation: getCallLegSessionInfo

Scope: Cluster

Displays information about a call leg in a call session.

- **id**: the CORBA reference to the call leg object in the Parlay Gateway (IpCallLeg).
- **callSessionId**: the ID of the call session.
- **callParticipantIdentifier**: the URI identifying the terminal associated with the call leg.
- **callParticipantStatus**: the status of the call participant. The status is one of:
  - **Initial**, during call setup.
  - **Connected**, during call duration.
  - **Terminated**, the participant has left the call.
- **callParticipantStartTime**: the time when the call participant was connected to the call.
- **callParticipantEndTime**: the time when the call participant was disconnected from the call.
- **callParticipantDuration**: the duration of the call for the participant. Given in seconds.



- `callParticipantTerminationCause`: the cause of the participant leaving the call. One of:
  - `Noanswer`, no answer from the participant.
  - `Busy`, the participant was busy (off-hook)
  - `Hangup`, the participant went on-hook.
  - `Notreachable`, could not reach the participant.
  - `Aborted`, the call was terminated for other reason than Hangup.
- `appInstGrpId`: the application instance associated with the application that created the call session.
- `callRef`: the CORBA reference to the call object in the Parlay Gateway (`IpMultiPartyCall`).
- `srcPlugin`: the type of plug-in instance that initiated the call. The type is one of:
  - `ThirdPartyCall`
  - `CallNotification`
  - `CallDirection`

Signature:

```
getCallLegSessionInfo(CallLegSessionId: String)
```

**Table 17-5** `getCallLegSessionInfo`

<code>getCallLegSessionInfo</code>	
Parameter	Description
<code>CallLegSessionId</code>	ID of the call leg session to get information about.

## Operation: `listCallSessionIds`

Scope: Cluster

Displays a list of IDs for ongoing call sessions. These are the Parlay X 3.0 `callSessionIDs`.

Signature:

```
listCallSessionIds()
```

**Table 17-6 listCallSessionIds**

listCallSessionIds	
Parameter	Description
-	-

## Operation: countPendingCallSession

Scope: Server

Displays the number of ongoing call sessions for this plug-in instance.

Signature:

```
countPendingCallSession()
```

**Table 17-7 countPendingCallSession**

countPendingCallSession	
Parameter	Description
-	-

## Parlay X 2.1 Third Party Call/INAP

This section contains a description of the configurations attributes and operations available for Parlay X 2.1 Third Party Call/INAP plug-in instance.

This plug-in instance uses the TietoEnator SS7 stack to connect to the network.

When configuring the SS7 connectivity, the settings in the management interface for the plug-in instance must be correlated with a sub-set of the settings in the following Stack-in-a-Box configuration files, see [Configuration files and dependencies](#).

Configuration and management of other parts of Stack-in-a-Box are outside the scope of this descriptio.Refer to the TietoEnator SS7 product documentation.

To see a	Refer to
Detailed list of necessary information for managing and configuring the plug-in instance	<a href="#">Properties for Parlay X 2.1 Third Party Call/INAP</a>
Configuration workflow	<a href="#">Configuration Workflow for Parlay X 2.1 Third Party Call/INAP</a>
List of operations and attributes related to management and provisioning	<a href="#">Management for Parlay X 2.1 Third Party Call/INAP</a>
Reference of management attributes and operations	<a href="#">Reference: Attributes and Operations for Parlay X 2.1 Third Party Call/INAP</a>

## Configuration Workflow for Parlay X 2.1 Third Party Call/INAP

1. Make sure the SS7 stack is configured and up and running. You must define an INAP user for each plug-in instance.
2. Using the Management Console or an MBean browser, select the MBean detailed in [Management for Parlay X 2.1 Third Party Call/INAP](#).
3. Configure connection information for the connection to the SS7 stack:
  - [Attribute: LocalSpc](#)
  - [Attribute: LocalSsn](#)
  - [Attribute: RemoteSpc](#)
  - [Attribute: RemoteSsn](#)
  - [Attribute: TSCFTimeout](#)
  - [Attribute: NoAnswerTimeout](#)
  - [Attribute: SccpPriority](#)
  - [Attribute: SccpQualityOfService](#)
  - [Attribute: InapUserId](#)

- [Attribute: Ss7Host](#)
- [Attribute: Ss7PortNumber](#)
- [Attribute: InapBindTimeout](#)

**Note:** When any of these attributes are changed, the [INAP API configuration file](#) is overwritten and the plug-in service must be restarted.

4. Set up the routing rules to the plug-in instance, see [Configuring the Plug-in Manager](#). Use the plug-in instance ID and address schemes detailed in [Properties for Parlay X 2.1 Third Party Call/INAP](#).

5. If desired, create and load a node SLA, see:

[Defining Global Node and Service Provider Group Node SLAs](#)

[Managing Application SLAs](#)

Move on to the provisioning of service provider accounts and application accounts.

## Properties for Parlay X 2.1 Third Party Call/INAP

Property	Description
Managed object in Administration Console	<domain name>->WLNG-><server name>->Communication Services->Plugin_third_party_call_inap
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName=Plugin_third_party_call_inap Type=com.bea.wlcp.wlng.plugin.tpc.inap.management.InapTpcMBean
Network protocol plug-in service ID	Plugin_third_party_call_inap
Network protocol plug-in instance ID	Plugin_px21_third_party_call_inap
Supported Address Scheme	tel
North interface	com.bea.wlcp.wlng.px21.plugin.ThirdPartyCallPlugin
Service type	ThirdPartyCall

Property	Description
Exposes to the service communication layer a Java representation of:	Parlay X 2.1 Part 2: Third Party Call
Interfaces with the network nodes using:	ETSI 94 INAP CS1, ETS 300 374-1
Deployment artifacts	tpc_inap.jar.jar, packaged in wlng_nt_third_party_call_px21.ear px21_third_party_call.war, packaged in wlng_at_third_party_call_px21.ear

This plug-in service does not support multiple instantiation using the Plug-in Manager. There is a one to one mapping between plug-in service and plug-in instance. The plug-in instance is created when the plug-in service is started.

## Management for Parlay X 2.1 Third Party Call/INAP

None.

## Reference: Attributes and Operations for Parlay X 2.1 Third Party Call/INAP

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

- [Attribute: LocalSpc](#)
- [Attribute: LocalSsn](#)
- [Attribute: RemoteSpc](#)
- [Attribute: RemoteSsn](#)
- [Attribute: TSCFTimeout](#)
- [Attribute: NoAnswerTimeout](#)
- [Attribute: SccpPriority](#)
- [Attribute: SccpQualityOfService](#)
- [Attribute: InapUserId](#)

- [Attribute: Ss7Host](#)
- [Attribute: Ss7PortNumber](#)
- [Attribute: InapBindTimeout](#)

## Attribute: LocalSpc

Scope: Server

Unit: n/a

Format: int [0-16383] or [0-16777215], depending on standard used.

Specifies the local SCCP Signaling Point Code (SPC) served by the local SS7 stack, that is, the SS7 network address for the plug-in instance. Used as the originating SPC by the plug-in instance.

Must be correlated with the property SCCP Local SPC in the back-end configuration file for the SS7 stack.

## Attribute: LocalSsn

Scope: Server

Unit: n/a

Format: int [2-254]

Specifies the local SCCP Sub System Number, to which the plug-in instance will bind itself.

Must be correlated with the property SCCP Local SSN in the back-end configuration file for the SS7 stack.

## Attribute: RemoteSpc

Scope: Server

Unit: n/a

Format: int [0-16383] or [0-16777215], depending on standard used.

Specifies the remote SCCP Signaling Point Code (SPC). Used in the destination address.

Must be correlated with the property SCCP Remote SPC in the back-end configuration file for the SS7 stack.

## Attribute: RemoteSsn

Scope: Server

Unit: n/a

Format: int [2-254]

Specifies the remote SCCP Signaling Sub System Number (SPC). Used in the destination address.

Must be correlated with the property SCCP Remote SSN in the back-end configuration file for the SS7 stack.

## Attribute: TSCFTimeout

Scope: Server

Unit: seconds

Format: int

Specifies the timeout value for the T(SCF) timer, used for supervising call establishment. It specifies how long the plug-in instance should wait for a response from an SCP after sending a request. If the time-out value is exceeded the TCAP dialogue is aborted.

## Attribute: NoAnswerTimeout

Scope: Server

Unit: seconds

Format: int [0-2047]

Specifies the time-out value for an INAP noAnswer event. The time-out value is used towards the signaling network in INAP DpSpecificCriteria when arming the noAnswer event.

## Attribute: SccpPriority

Scope: Server

Unit: n/a

Format: int [0-3]

Specifies the SCCP priority indicator. 0 is the lowest priority and 3 is the highest priority.

## Attribute: SccpQualityOfService

Scope: Server

Unit: n/a

Format: int [0-3]

Specifies the SCCP quality of service indicator.

## Attribute: InapUserId

Scope: Server

Unit: n/a

Format: String

Specifies the user ID used by the INAP plug-in when connecting to the SS7 stack. Must be defined in the common parts configuration file, see [Common parts configuration file](#).

## Attribute: Ss7Host

Scope: Server

Unit: n/a

Format: String

Specifies the host name or IP address of the SS7 stack. Separate the host names or IP addresses with a comma (,) if the stack is running in HA mode. Also if you are using HA mode, use this attribute to define the port number to use: for example 192.168.0.19:99, and do not use [Attribute: Ss7PortNumber](#) to specify the port number.

## Attribute: Ss7PortNumber

Scope: Server

Unit: n/a

Format: int

Specifies the port number to use in connecting to the stack.



## Attribute: InapBindTimeout

Scope: Server

Unit: milliseconds

Format: int

Specifies the stack bind timeout value.

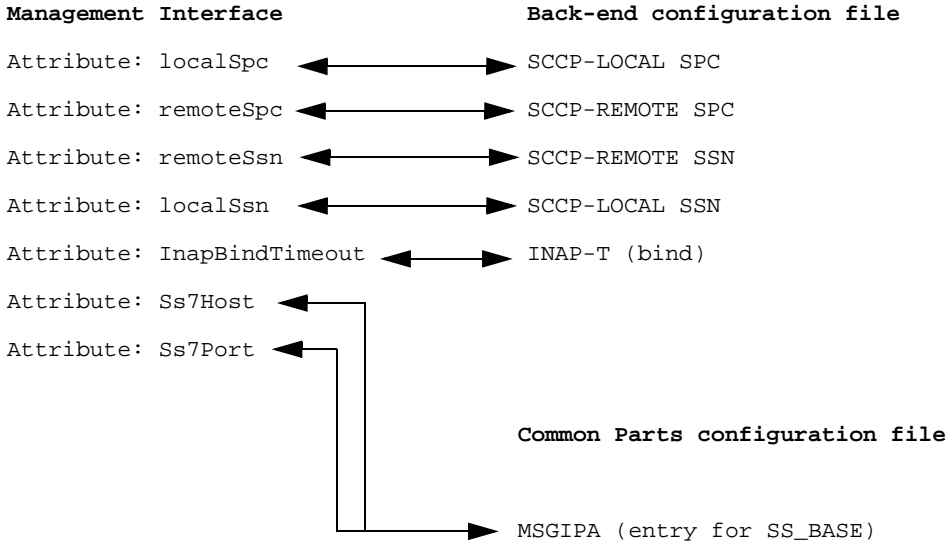
## Configuration files and dependencies

There are a set of dependencies and settings that must be correlated between the configuration files and the configuration settings in the plug-in instance. The following files have touch-points:

- INAP API configuration file (`Plugin_px21_third_party_call_inap.properties`): see [INAP API configuration file](#).
- SS7 back-end configuration file (`itu_ss7.cfg`): see [Back-end configuration file](#).
- Common parts configuration file (`cp.cnf`): see [Common parts configuration file](#).

The specific settings are explained in the sections describing the settings in the management interface for the plug-in instance and the description of the files. [Figure 17-1](#) below presents an overview of the dependencies.

**Figure 17-1 Dependencies between settings in the plug-in instance and properties in the configuration files for the SS7 stack**



## INAP API configuration file

The INAP API configuration file is a configuration file for the TietoEnator JAIN INAP API library used by the plug-in instance.

This file provides the API with information on where to connect, how to bind to the stack, and values for some parameters that are not exposed in the API. If any of the properties are not set, default values are used.

The properties that are related to the interface between the plug-in instance and the stack are described in [Table 17-9](#). See the documentation for the stack for a full description of all settings.

The file is located on the file system of the host where the plug-in instance is running; in default installations this is in the domain home directory of Network Gatekeeper. The file is created and updated when an attribute is updated using the MBean for the Parlay x 2.1 Third Party Call/INAP plug-in. The plug-in service needs to be restarted for the changes to effect.

The file is named `Plugin_px21_third_party_call_inap.properties` and it is located in `$DOMAIN_HOME`.

Any changes to the MBean attributes causes the file to be overwritten, and hence any modifications to it are lost.

**Table 17-8 INAP API properties used**

Property	Comments
priority	SCCP Message priority.
quality-of-service	SCCP QoS.
inap-user-id	<p>The common parts module ID used by the plug-in instance. See <a href="#">Common parts configuration file</a>.</p> <p>Either INAPUP or any of the USERxx IDs are to be used by the plug-in instance.</p> <p>The numeric identifier of the ID shall be used, not the ID itself (as used in the common parts configuration file).</p> <p>The module ID numeric values can be found in <code>/opt/ss7/ss7_ITU/include/portss7.h</code> in an installed stack.</p> <p>USER01 has decimal value 40, USER02 41, and so on.</p>
ss7host	<p>The host name or IP-address of the host running the SS7 back-end, that is, the address to which the SS7_BASE module ID is bound in the common parts configuration file. See <a href="#">Common parts configuration file</a>.</p> <p>If several SS7 back-ends are used, either in high-availability mode or horizontally distributed mode, enter the host name (or IP-address) for the servers in a comma separated list.</p>
port-number	The port number to which the SS7_BASE module ID is bound in the common parts configuration file.

**Table 17-8 INAP API properties used**

Property	Comments
bind-timeout	<p>Time to wait for bind response before a bind operation is considered a failure.</p> <p>Unit: milliseconds.</p>
heartbeat-interval	<p>Heartbeat interval between the INAP API and the common parts module.</p> <p>Must correspond to the MSGHBRATE and MSGHBLOST properties defined in the common parts configuration file. See <a href="#">Common parts configuration file</a>.</p> <p><b>Note:</b> This property is not generated from the settings in the MBean of the Parlay X 2.1 Third Party Call/INAP plug-in. The absence of the property means that no heartbeats are sent. If heartbeats are used, this property must be added manually in the configuration file. If used the recommended value is MSGHBRATE times MSGHBLOST. Any changes to the MBean attributes causes the file to be overwritten, and hence this setting is lost.</p> <p>Unit: milliseconds.</p>

**Listing 17-1 Example INAP JAIN API Configuration file**

```

local-ssn: 254
priority: 0
quality-of-service: 0
trace-level: 0
inap-user-id: 40
ss7host: 192.168.20.1,192.168.20.2
port-number: 7001
bind-timeout: 5000
    
```

## Common parts configuration file

The SS7 Common parts configuration file is a configuration file specifying the interprocess communication for the SS7 stack, including users of the stack. The plug-in instance acts as user of the stack via the INAP API. The dependencies to the plug-in instance and the stack is described in [Table 17-9](#). All other settings are related to the stack itself: see the documentation for the stack for a description of these settings.

The file is located on the file system of the host running the back-end part of the SS7 stack. In default installations this is in `/opt/ss7/ss7_ITU/etc/cp.cnf`.

**Table 17-9 Common parts configuration file properties with dependencies to plug-in instance settings**

Property	Comments
MSGIPA	<p>There must be one MSGIPA entry per plug-in instance.</p> <p>First, choose a Message Port owner ID (MP OwnerID). Use one of the following:</p> <ul style="list-style-type: none"> <li>• INAPUP</li> <li>• USER01</li> <li>• USER02</li> <li>• USER03</li> <li>• USER04</li> <li>• USER05</li> <li>• USER06</li> <li>• USER07</li> <li>• USER08</li> <li>• USER09</li> <li>• USER10</li> </ul> <p>MP OwnerID shall correspond to inap-user-id specified in <a href="#">Attribute: InapUserId</a></p> <p>The IP-address (or host name) with TCP port number must correspond to the host where the plug-in instance is deployed.</p> <p>Make sure there is a MSGINTERACT entry per MP OwnerID.</p> <p>Instances of MP OwnerIDs are not supported.</p> <p>Example:</p> <p>MSGIPA=USER01,192.168.20.2:6701 10.10.10.11:6701</p>
MSGHBLOST	<p>Must correspond to <a href="#">Attribute: LocalSpC</a>.</p>
MSGHBRATE	<p>Must correspond to heartbeat-interval in <a href="#">INAP API configuration file</a></p>

## Back-end configuration file

The back-end configuration file contains the configuration of the SS7 back-end stack layers. Each stack layer has a dedicated section in this file, and it is where, for example, SS7 network routing and protocol timers are configured. The dependencies between the plug-in instance and the stack

are described in [Table 17-9](#). All other settings are related to the stack itself: see the documentation for the stack for a description of these settings.

The file is located on the file system of the host running the back-end part of the SS7 stack, in default installations this is in `/opt/ss7/ss7_ITU/etc/ss7_itu.cnf`

**Table 17-10 Back-end configuration file properties with dependencies to plug-in instance settings**

Property	Comments
INAP-T (bind)	Must correspond to <a href="#">Attribute: InapBindTimeout</a> given in <code>Plugin_third_party_call_inap</code> .
SCCP-LOCAL SPC	Must correspond to <a href="#">Attribute: LocalSpc</a> given in <code>Plugin_third_party_call_inap</code> .
SCCP-LOCAL SSN	Must correspond to <a href="#">Attribute: LocalSsn</a> given in <code>Plugin_third_party_call_inap</code> .
SCCP-REMOTE SPC	Must correspond to <a href="#">Attribute: RemoteSpc</a> given in <code>Plugin_third_party_call_inap</code> .
SCCP-REMOTE SSN	Must correspond to <a href="#">Attribute: RemoteSsn</a> given in <code>Plugin_third_party_call_inap</code> .

## Parlay X 2.1 Third Party Call/SIP

This section contains a description of the configuration attributes and operations available for the Parlay X 2.1 Third Party Call/SIP protocol translation module.

Parlay X 2.1 Third Party Call/SIP uses two parts for SIP connectivity, a part that executes a network protocol plug-in instance in Network Gatekeeper container, and a part that executes as a SIP application in the SIP Server container.

To see a	Refer to
Detailed list of necessary information for managing and configuring the plug-in instance	<a href="#">Properties for Parlay X 2.1 Third Party Call/SIP</a>
Configuration workflow	<a href="#">Configuration Workflow for Parlay X 2.1 Third Party Call/SIP</a>
List of operations and attributes related to management and provisioning	<a href="#">Management for Parlay X 2.1 Third Party Call/SIP</a>
Reference of management attributes and operations	<a href="#">Reference: Attributes and Operations for Parlay X 2.1 Third Party Call/SIP</a>

## Configuration Workflow for Parlay X 2.1 Third Party Call/SIP

1. Using the Management Console or an MBean browser, select the MBean detailed in [Properties for Parlay X 2.1 Third Party Call/SIP](#).
2. Configure behavior of the network protocol plug-in instance:
  - [Attribute: ChargingAllowed](#)
  - [Attribute: StatusRetentionTime](#)
3. Configure connection information to the SIP server:
  - [Attribute: SIPNodeUrl](#)
  - [Attribute: SIPNodeUsername](#)
  - [Attribute: SIPNodePassword](#)
4. Configure connection information from the SIP server to Gatekeeper, see [Configuring WebLogic SIP Server Integration](#).
5. Set up the routing rules to the plug-in instance, see [Configuring the Plug-in Manager](#). Use the plug-in instance ID and address schemes detailed in [Properties for Parlay X 2.1 Third Party Call/SIP](#).
6. If desired, create and load a node SLA, see:  
[Defining Global Node and Service Provider Group Node SLAs](#)



### Managing Application SLAs

Move on to the provisioning of service provider accounts and application accounts.

## Management for Parlay X 2.1 Third Party Call/SIP

None.

### Properties for Parlay X 2.1 Third Party Call/SIP

Property	Description
Managed object in Administration Console	<domain name>->WLNG-><server name>->Communication Services->Plugin_third_party_call_sip
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName=Plugin_third_party_call_sip Type=com.bea.wlcp.wlng.plugin.tpc.sip.TPCMBean
Network protocol plug-in service ID	Plugin_third_party_call_sip
Network protocol plug-in instance ID	Plugin_third_party_call_sip
Supported Address Scheme	sip
North interface	com.bea.wlcp.wlng.px21.plugin.ThirdPartyCallPlugin
Service type	ThirdPartyCall
Exposes to the service communication layer a Java representation of:	Parlay X 2.1 Part 2: Third Party Call

Property	Description
Interfaces with the network nodes using:	SIP: Session Initiation Protocol, RFC 3261
Deployment artifacts	Plugin_px21_third_party_call_sip.jar, packaged in wlng_nt_third_party_call_px21.ear px21_third_party_call.war, packaged in wlng_at_third_party_call_px21.ear wlss_tpc_ejb.jar and wlss_tpc_web.war, packaged in wlss-int.ear

This plug-in service does not support multiple instantiation using the Plug-in Manager. There is a one to one mapping between plug-in service and plug-in instance. The plug-in instance is created when the plug-in service is started.

## Reference: Attributes and Operations for Parlay X 2.1 Third Party Call/SIP

Below is a list of attributes for configuration and maintenance:

- [Attribute: ChargingAllowed](#)
- [Attribute: SIPNodeUrl](#)
- [Attribute: SIPNodeUsername](#)
- [Attribute: SIPNodePassword](#)
- [Attribute: StatusRetentionTime](#)

### Attribute: ChargingAllowed

Scope: Cluster

Unit: n/a

Format: boolean

Specifies whether charging is allowed. That is, if the charging parameter can be present in a request from an application.

## Attribute: SIPNodeUrl

Scope: Cluster

Unit: n/a

Format: String in URL format

Specifies the WebLogic SIP Server JNDI URL. This was configured when WebLogic SIP Server was installed.

WebLogic SIP Server naming service contains the host name (or IP address) and the port number identifying `wlss.jndi.uri`.

Example: `t3://127.0.0.1:7002`

**Note:** This is not the same as the Network Gatekeeper naming service.

## Attribute: SIPNodeUsername

Scope: Cluster

Unit: n/a

Format: String

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.

## Attribute: SIPNodePassword

Scope: Cluster

Unit: n/a

Format: String

Specifies the password associated with [Attribute: SIPNodeUsername](#).

## Attribute: StatusRetentionTime

Scope: Cluster

Unit: minutes

Format: int

Specifies the length of time to retain status information about the call after it has been terminated.

## Managing and Configuring Third Party Call Communication Services

# Managing and Configuring Call Notification Communication Services

The following section describes configuration and maintenance attributes and operations for the Parlay X Call Notification communication services:

- [Parlay X 3.0 Call Notification/Parlay MultiParty Call Control](#)
  - [Configuration Workflow for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control](#)
  - [Management and Provisioning Operations for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control](#)
  - [Reference: Attributes and Operations for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control](#)
- [Parlay X 2.1 Call Notification/SIP](#)
  - [Configuration Workflow for Parlay X 2.1 Call Notification/SIP](#)
  - [Reference: Attributes and Operations Parlay X 2.1 Call Notification/SIP](#)

## Parlay X 3.0 Call Notification/Parlay MultiParty Call Control

This section contains a description of the configuration attributes and operations available for the Parlay X 3.0 Call Notification/Parlay MultiParty Call Control network protocol plug-in instance

Most of the configuration is done in the OSA Access module, but with configuration parameters for Parlay MultiParty Call Control. See [Managing OSA/Parlay Gateway Connections using Parlay\\_Access](#).

Two different types of notifications are managed:

- Regular notifications, started by an application invoking `startCallNotification` or `startCallDirectionNotification`
- Media notifications, started by an application invoking `startPlayAndCollectNotification`

To see a	Refer to
Detailed list of necessary information for managing and configuring the plug-in instance	<a href="#">Properties for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control</a>
Configuration workflow	<a href="#">Configuration Workflow for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control</a>
List of operations and attributes related to management and provisioning	<a href="#">Management and Provisioning Operations for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control</a>
Reference of management attributes and operations	<a href="#">Reference: Attributes and Operations for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control</a>

## Configuration Workflow for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control

1. Using the Management Console or an MBean browser, select the MBean detailed in [Properties for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control](#).
2. Gather information about the OSA/Parlay Gateway and configure the protocol translator accordingly. The following information needs to be obtained from the OSA/Parlay Gateway administrator and configured in the Parlay 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.
3. Set up the OSA/Parlay Client and the OSA/Parlay Client Mappings according to [Mapping the OSA client to an OSA Gateway and an OSA/Parlay SCS](#) in section [Managing OSA/Parlay Gateway Connections using Parlay\\_Access](#).
  4. Set up the routing rules to the plug-in instance, see [Configuring the Plug-in Manager](#). Use the plug-in instance ID and address schemes detailed in [Properties for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control](#).
  5. If desired, create and load a node SLA, see:
    - [Defining Global Node and Service Provider Group Node SLAs](#)
    - [Managing Application SLAs](#)
- Move on to the provisioning of service provider accounts and application accounts.

## Properties for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control

Property	Description
Managed object in Administration Console	<domain name>->WLNG-><server name>->Communication Services->Plugin_px30_call_notification_parlay_mpcc
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName=Plugin_px30_cn_parlay Type=com.bea.wlcp.wlng.px30.plugin.callnotification.parlay.management.mbean.CallNotificationMBean
Network protocol plug-in service ID	Plugin_px30_call_notification_parlay_mpcc
Network protocol plug-in instance ID	Plugin_px30_call_notification_parlay_mpcc
Supported Address Scheme	tel
North interfaces	com.bea.wlcp.wlng.px30.plugin.CallNotificationManagerPlugin com.bea.wlcp.wlng.px30.plugin.CallDirectionManagerPlugin com.bea.wlcp.wlng.px30.callback.CallDirectionCallback com.bea.wlcp.wlng.px30.callback.CallNotificationCallback
Service type	CallNotification
Exposes to the service communication layer a Java representation of:	Parlay X 3.0 Part 3: Call Notification
Interfaces with the network nodes using:	Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control SCF; Subpart 7: MultiParty Call Control Service
Deployment artifacts	px30_callnotification_parlay.jar, packaged in wlng_nt_call_notification_px30.ear  px30_call_notification.war and px30_call_notification_callback.jar, packaged in wlng_at_call_notification_px30.ear



This plug-in service does not support multiple instantiation using the Plug-in Manager. There is a one to one mapping between plug-in service and plug-in instance. The plug-in instance is created when the plug-in service is started.

## Management and Provisioning Operations for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control

The following operations are related to provisioning and management:

- [Operation: deleteMediaNotification](#)
- [Operation: deleteNotification](#)
- [Operation: getMediaNotification](#)
- [Operation: getNotification](#)
- [Operation: listNotifications](#)
- [Operation: listMediaNotifications](#)

## Reference: Attributes and Operations for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control

Managed object: Communication Services->Plugin\_px30\_call\_notification\_parlay\_mpcc

MBean:

com.bea.wlcp.wlmg.px30.plugin.callnotification.parlay.management.mbean.CallNotificationMBean

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

- [Operation: deleteMediaNotification](#)
- [Operation: deleteNotification](#)
- [Operation: getMediaNotification](#)
- [Operation: getNotification](#)
- [Operation: listNotifications](#)
- [Operation: listMediaNotifications](#)

## Operation: deleteMediaNotification

Scope: Cluster

Deletes a media notification.

Signature:

```
deleteMediaNotification(correlator: String)
```

Table 18-1 deleteMediaNotification

---

deleteNotification	
Parameter	Description
correlator	ID for the subscription. Given by an application when the subscription was started.

---

## Operation: deleteNotification

Scope: Cluster

Deletes a notification from the storage and removes it from OSA Gateway.

Signature:

```
deleteNotification(correlator: String)
```

Table 18-2 deleteNotification

---

deleteNotification	
Parameter	Description
correlator	ID for the subscription. Given by an application when the subscription is started.

---

## Operation: getMediaNotification

Scope: Cluster

Displays information about a media notification. The information includes:

- Parlay X correlator.
  - Parlay X callSessionIdentifier
  - needNotify -internal field. If true, Parlay X 3.0 Audio Call for MPCC and CUI plug-in instance needs to invoke the Parlay X 3.0 Call Notification for MPCC callback method sendInfoAndCollectRes.
  - Parlay IpAppUICallRef (CORBA IOR)
  - Parlay X endPoint to where the notification is sent
  - Data about the owner of the notification:
    - Service provider account ID
    - Application account ID
    - Application instance ID
    - notifMode, used to distinguish which kind of media notification the notification belongs to. That is, which Parlay X operation that created the notification:
      - 1 -the notification was created using startPlayAndCollectInteraction.
      - 2 -the notification was created using startPlayAndRecordInteraction.
- Note:** This operation is not supported in Network Gatekeeper 4.0

Signature:

```
getMediaNotification(correlator: String)
```

**Table 18-3 getMediaNotification**

<b>getMediaNotification</b>	
<b>Parameter</b>	<b>Description</b>
correlator	ID for the subscription. Given by an application when the subscription is started.

## Operation: getNotification

Scope: Cluster

Displays information about a notification. The information includes:

- Parlay X Correlator.
- Parlay X Endpoint where notifications are sent.
- List of Parlay `notificationIds` associated with the Parlay X notification. There is one for each called party address for which notifications should be triggered supplied in the Parlay X operations `startCallNotification` or `startCallDirectionNotification`.
- Data about the owner of the notification:
  - Service provider account ID
  - Application account ID
  - Application instance ID

Signature:

```
getNotification(correlator: String)
```

**Table 18-4** `getNotification`

<code>getNotification</code>	
Parameter	Description
<code>correlator</code>	ID of the notification. Given by an application when the notification is started.

## Operation: `listNotifications`

Scope: Cluster

Displays all active call notifications. That is, notifications registered by an application using:

- Operation: `startCallDirectionNotification`
- Operation: `startCallNotification`

Returns a list of correlators that uniquely identifies each notification.

Signature:

```
listNotifications()
```

**Table 18-5 listNotifications**

listNotifications	
Parameter	Description
-	-

## Operation: listMediaNotifications

Scope: Cluster

Displays all active media notifications. That is, notifications that have been registered by an application using:

- Operation: startPlayAndCollectNotification
- Operation: startPlayAndRecordNotification

**Note:** This operation is not supported in Network Gatekeeper 4.0

Returns a list of correlators that uniquely identify each notification.

Signature:

```
listMediaNotifications()
```

**Table 18-6 listMediaNotifications**

listNotifications	
Parameter	Description
-	-

## Parlay X 2.1 Call Notification/SIP

This section contains a description of the configurations attributes and operations available for the Parlay X 2.1 Call Notification/SIP plug-in instance.

Parlay X 2.1 Call Notification/SIP uses two parts for SIP connectivity, a part that executes as a network protocol plug-in instance in Network Gatekeeper container, and a part that executes as a

SIP application in the SIP Server container. The two parts execute in different containers and must be configured in both.

To see a	Refer to
Detailed list of necessary information for managing and configuring the plug-in	<a href="#">Properties for Parlay X 2.1 Call Notification/SIP</a>
Configuration workflow	<a href="#">Properties for Parlay X 2.1 Call Notification/SIP</a>
List of operations and attributes related to management and provisioning	<a href="#">Management for Parlay X 2.1 Call Notification/SIP</a>
Reference of management attributes and operations	<a href="#">Reference: Attributes and Operations Parlay X 2.1 Call Notification/SIP</a>

## Properties for Parlay X 2.1 Call Notification/SIP

Property	Description
Managed object in Administration Console	<domain name>->WLNG-><server name>->Communication Services->Plugin_call_notification_sip
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName=Plugin_call_notification_sip Type=com.bea.wlcp.wlng.plugin.callnotification.sip.impl.CallNotificationMBeanImpl
Network protocol plug-in service ID	Plugin_px21_call_notification_sip
Network protocol plug-in instance ID	Plugin_px21_call_notification_sip
Supported Address Scheme	sip

Property	Description
North interfaces	com.bea.wlcp.wlng.px21.plugin.CallNotificationManagerPlugin om.bea.wlcp.wlng.px21.plugin.CallDirectionManagerPlugin
Service type	CallNotification
Exposes to the service communication layer a Java representation of:	Parlay X 2.1 Part 3: Call Notification
Interfaces with the network nodes using:	SIP: Session Initiation Protocol, RFC 3261
Deployment artifacts	callnotification_sip.jar, plug_in_call_notification_callback_ejb.jar, and px21_call_notification_service.jar packaged in wlng_nt_call_notification_px21.ear  px21_call_notification.war and px21_call_notification_callback.jar, packaged in wlng_at_call_notification_px21.ear  wlss-call-notification.war, packaged in wlss-int.ear

This plug-in service does not support multiple instantiation using the Plug-in Manager. There is a one to one mapping between plug-in service and plug-in instance. The plug-in instance is created when the plug-in service is started.

## Configuration Workflow for Parlay X 2.1 Call Notification/SIP

1. In Network Gatekeeper Management Console, set up the routing rules to the plug-in instance: see [Configuring the Plug-in Manager](#). Use the plug-in instance ID and address schemes detailed in [Properties for Parlay X 2.1 Call Notification/SIP](#).
2. Configure connection information from the SIP server to Gatekeeper, see [Configuring WebLogic SIP Server Integration](#).
3. If desired, create and load a node SLA, see:
  - [Defining Global Node and Service Provider Group Node SLAs](#)
  - [Managing Application SLAs](#)

Move on to the provisioning of service provider accounts and application accounts.

## Management for Parlay X 2.1 Call Notification/SIP

The following operations are related to management:

- [Operation: getCallDirectionSubscription](#)
- [Operation: getNotificationSubscription](#)
- [Operation: listCallDirectionSubscriptions](#)
- [Operation: listNotificationSubscriptions](#)
- [Operation: removeAllCallDirectionSubscriptions](#)
- [Operation: removeAllNotificationSubscriptions](#)
- [Operation: removeCallDirectionSubscription](#)
- [Operation: removeNotificationSubscription](#)

## Reference: Attributes and Operations Parlay X 2.1 Call Notification/SIP

Below is a list of attributes and operations for configuration 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 18-7 getCallDirectionSubscription**

<b>getCallDirectionSubscription</b>	
<b>Parameter</b>	<b>Description</b>
Correlator	ID for the subscription. Given by an application when the subscription was started.

## Operation: getNotificationSubscription

Scope: Cluster

Displays call notification subscription information.

Signature:

```
getNotificationSubscription(Crrelator: String)
```

**Table 18-8 getCallDirectionSubscription**

<b>getCallDirectionSubscription</b>	
<b>Parameter</b>	<b>Description</b>
Correlator	ID for the subscription. Given by an application when the subscription was started.

## Operation: listCallDirectionSubscriptions

Scope: Cluster

Displays a list of correlators for call direction subscriptions.

Signature:

```
listCallDirectionSubscriptions(Offset: int, Length: int)
```

**Table 18-9 listCallDirectionSubscriptions**

<b>listCallDirectionSubscriptions</b>	
<b>Parameter</b>	<b>Description</b>
Offset	Start of offset.
Length	Number of entries returned.

## Operation: listNotificationSubscriptions

Scope: Cluster

Displays a list of correlators for call notification subscriptions.

Signature:

```
listNotificationSubscriptions(Offset: int, Length: int)
```

**Table 18-10 listNotificationSubscriptions**

<b>listNotificationSubscriptions</b>	
<b>Parameter</b>	<b>Description</b>
Offset	Start of offset.
Length	Number of entries returned.

## Operation: removeAllCallDirectionSubscriptions

Scope: Cluster

Removes all call direction subscriptions.

Signature:

```
removeAllCallDirectionSubscriptions()
```

**Table 18-11** `removeAllCallDirectionSubscriptions`

<code>removeAllCallDirectionSubscriptions</code>	
Parameter	Description
-	-

## Operation: `removeAllNotificationSubscriptions`

Scope: Cluster

Removes all call notification subscriptions.

Signature:

```
removeAllNotificationSubscriptions()
```

**Table 18-12** `removeAllNotificationSubscriptions`

<code>removeAllNotificationSubscriptions</code>	
Parameter	Description
-	-

## Operation: `removeCallDirectionSubscription`

Scope: Cluster

Removes a call direction subscription.

Signature:

```
removeCallDirectionSubscription(Correlator: String)
```

**Table 18-13** `removeCallDirectionSubscription`

---

**removeCallDirectionSubscription**

---

Parameter	Description
Correlator	ID for the subscription. Given by an application when the subscription was started.

---

## Operation: `removeNotificationSubscription`

Scope: Cluster

Removes a call notification subscription.

Signature:

```
removeNotificationSubscription(Correlator: String)
```

**Table 18-14** `removeNotificationSubscription`

---

**removeNotificationSubscription**

---

Parameter	Description
Correlator	ID for the subscription. Given by an application when the subscription was started.

---

# Managing and Configuring Short Messaging Communication Services

The following section describes configuration and maintenance attributes and operations for the communication services that expose Parlay X 2.1 Short Messaging and Extended Web Services Binary SMS Web Services interfaces. The sections also provide a workflow for the configuration:

- [Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP](#)
  - [Properties for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP](#)
  - [Configuration Workflow for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP](#)
  - [Management Operations for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP](#)
  - [Reference: Attributes and Operations for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP](#)

## Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP

This section contains a description of the configuration attributes and operations available for the Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP network protocol plug-in instances. The plug-in instance is shared between these.

To see a	Refer to
Detailed list of necessary information for managing and configuring the plug-in instance	<a href="#">Properties for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP</a>
Configuration workflow	<a href="#">Configuration Workflow for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP</a>
List of operations and attributes related to management and provisioning	<a href="#">Management Operations for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP</a>
Reference of management attributes and operations	<a href="#">Reference: Attributes and Operations for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP</a>

## Properties for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP

Property	Description
Managed object in Administration Console	<domain name>->WLNG-><server name>->Communication Services-><plug-in instance ID>
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName is same as plug-in instance ID. Type=com.bea.wlcp.wlng.plugin.sms.smpp.management.SmsMBean
Network protocol plug-in service ID	Plugin_px21_short_messaging_smpp

Property	Description
Network protocol plug-in instance ID	The ID is given when the plug-in instance is created, see <a href="#">Managing and Configuring the Plug-in Manager</a> .
Supported Address Scheme	tel
North interfaces	<p>Parlay X 2.1 Short Messaging/SMPP:</p> <ul style="list-style-type: none"> <li>com.bea.wlcp.wlng.px21.plugin.SendSmsPlugin</li> <li>com.bea.wlcp.wlng.px21.plugin.ReceiveSmsPlugin</li> <li>com.bea.wlcp.wlng.px21.plugin.SmsNotificationManagerPlugin</li> <li>com.bea.wlcp.wlng.px21.callback.SmsNotificationCallback</li> </ul> <p>Extended Web Services Binary SMS/SMPP:</p> <ul style="list-style-type: none"> <li>com.bea.wlcp.wlng.ews.plugin.BinarySmsPlugin</li> </ul>
Service type	<p>Parlay X 2.1 Short Messaging/SMPP:</p> <ul style="list-style-type: none"> <li>Sms</li> </ul> <p>Extended Web Services Binary SMS/SMPP:</p> <ul style="list-style-type: none"> <li>BinarySms</li> </ul>
Exposes to the service communication layer a Java representation of:	<p>Parlay X 2.1 Short Messaging/SMPP:</p> <ul style="list-style-type: none"> <li>Parlay X 2.1 Part 4: Short Messaging</li> </ul> <p>Extended Web Services Binary SMS/SMPP:</p> <ul style="list-style-type: none"> <li>Extended Web Services Binary SMS</li> </ul>
Interfaces with the network nodes using:	SMPP 3.4
Deployment artifacts	<p>Parlay X 2.1 Short Messaging/SMPP:</p> <p>Plugin_px21_short_messaging.jar, and px21_sms_service.jar packaged in wlng_nt_sms_px21.ear</p> <p>px21_sms.war and px21_sms_callback.jar, packaged in wlng_at_sms_px21.ear</p> <p>Extended Web Services Binary SMS/SMPP:</p> <p>ews_binary_sms_service.jar packaged in wlng_nt_sms_px21.ear</p> <p>ews_binary_sms.war packaged in wlng_at_sms_px21.ear</p>

## Configuration Workflow for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP

Below is an outline of configuring a plug-in instance using the Network Gatekeeper Management Console:

1. Create one or more instances of the plug-in service: see [Managing and Configuring the Plug-in Manager](#). Use the plug-in service ID as detailed in [Properties for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP](#).
2. Using the Management Console or an MBean browser, select the MBean for the plug-in instance. The MBean display name is the same as the plug-in instance ID given when the plug-in instance was created.
3. Configure the behavior of the plug-in instance:
  - Attribute: [DefaultDataCoding](#)
  - Attribute: [EsmeAddressRange](#)
  - Attribute: [EsmeNpi](#)
  - Attribute: [EsmePassword](#)
  - Attribute: [EsmeSystemId](#)
  - Attribute: [EsmeSystemType](#)
  - Attribute: [EsmeTon](#)
  - Attribute: [ForwardXParams](#)
  - Attribute: [RequestDeliveryReports](#)
  - 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](#)
  - Attribute: [SegmentsLimit](#)
4. Specify [Attribute: BindType](#), and depending on which type, the following:
- [Attribute: TransmitterProxyLocalAddressAndPort](#), [Attribute: ReceiverProxyLocalAddressAndPort](#), [Attribute: NumberTransmitterConnections](#), and [Attribute: NumberReceiverConnections](#)
- or
- [Attribute: TransceiverProxyLocalAddressAndPort](#) and [Attribute: NumberTransceiverConnections](#).
5. Specify [Attribute: WindowSize](#), and depending on the setting, the following:
- [Attribute: WindowWaitTimeout](#)
6. Specify [Attribute: ReceiveSmsIgnoreMissingSegments](#), and depending on the setting, the following:
- [Attribute: ReceiveSegmentsWaitTime](#)
7. Set up the routing rules to the plug-in instance: see [Configuring the Plug-in Manager](#). Use the plug-in instance ID and address schemes detailed in [Properties for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP](#)
8. If desired, create and load a node SLA, see:
- [Defining Global Node and Service Provider Group Node SLAs](#)
  - [Managing Application SLAs](#)

Continue with the provisioning of service provider accounts and application accounts.

## Management Operations for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP

The following operations are related to management:

- Operation: [enableReceiveSms](#)
- Operation: [countOfflineNotificationCache](#)
- Operation: [countOnlineNotificationCache](#)
- Operation: [countSmsCache](#)
- Operation: [getOfflineNotificationInfo](#)
- Operation: [getOnlineNotificationInfo](#)
- Operation: [removeOfflineNotificationInfo](#)
- Operation: [removeOnlineNotificationInfo](#)
- Operation: [resetSMPPConnection](#)

## Reference: Attributes and Operations for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP

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

- Attribute: [ActiveStatus \(r\)](#)
- Attribute: [BindType](#)
- Attribute: [ConnectDelayValue](#)
- Attribute: [ConnectTotalTimeValue](#)
- Attribute: [DefaultDataCoding](#)
- Attribute: [MobileCountryCode](#)
- Attribute: [MobileNetworkCode](#)

- Attribute: NumberReceiverConnections
- Attribute: NumberTransceiverConnections
- Attribute: NumberTransmitterConnections
- Attribute: EnquireLinkRequestTimerValue
- Attribute: EnquireLinkTimerValue
- Attribute: EnquireLinkTimerValue
- Attribute: EsmeNpi
- Attribute: EsmeTon
- Attribute: EsmeAddressRange
- Attribute: EsmeSystemType
- Attribute: EsmeSystemId
- Attribute: EsmePassword
- Attribute: ForwardXParams
- Attribute: NumberReceiverConnections
- Attribute: NumberTransceiverConnections
- Attribute: NumberTransmitterConnections
- Attribute: RequestDeliveryReports
- Attribute: ReceiveSegmentsWaitTime
- Attribute: RequestTimerValue
- Attribute: SequenceNumberRangeEndId
- Attribute: SequenceNumberRangeStartId
- Attribute: SMSCDefaultAlphabet
- Attribute: SmppVersion (r)
- Attribute: SmscPort
- Attribute: SmscAddress

- Attribute: RequestDeliveryReports
- Attribute: TransceiverProxyLocalAddressAndPort
- Attribute: TransmitterProxyLocalAddressAndPort
- Attribute: UserTextMaxLength
- Attribute: WindowSize
- Attribute: WindowWaitTimeout
- Operation: addOriginatingAddressTypeMapping
- Operation: countOfflineNotificationCache
- Operation: countOnlineNotificationCache
- Operation: countSmsCache
- Operation: enableReceiveSms
- Operation: getOfflineNotificationInfo
- Operation: getOnlineNotificationInfo
- Operation: listOfflineNotificationInfo
- Operation: listOnlineNotificationInfo
- Operation: listOriginatingAddressTypeMappings
- Operation: removeOfflineNotificationInfo
- Operation: removeOnlineNotificationInfo
- Operation: removeOriginatingAddressTypeMapping
- 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 instance and the SMSC. True if the transmitter is successfully connected to the SMPP server; false if not.

## Attribute: **BindType**

Scope: Server

Unit: n/a

Format: int enumeration [0, 1]

Specifies how the plug-in shall bind to the SMSC.

Use:

- **0** to bind as TRANSMITTER and RECEIVER
- **1** to bind as Transceiver

## 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 instance 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 SMSes to be sent to the plug-in instance 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 instance 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 instance 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 instance when connecting to the SMSC as an ESME.

## Attribute: EsmeTon

Scope: Server

Unit: n/a

Format: Integer

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: ForwardXParams

Scope: Server

Unit: n/a

Format: Boolean

Specifies if tunnelled parameters are forwarded to the plug-in instance or not. Use:

- **true** to enable forwarding
- **false** to disable forwarding

## Attribute: ReceiveSmsIgnoreMissingSegments

Scope: Server

Unit: n/a

Format: Boolean [TRUE | FALSE]

Specifies if the plug-in instance shall deliver network-triggered short messages with missing message segments to applications or not.

Use:

- **true** if the plug-in shall assemble received segments and deliver the incomplete message to the application.
- **false** if the plug-in shall not deliver messages to the application unless all segments are received.

## Attribute: RequestDeliveryReports

Scope: Cluster

Unit: n/a

Format: Boolean [TRUE | FALSE]

Specifies if the default behavior of the plug-in instance 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 ability 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 instance 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

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: NumberReceiverConnections

Scope: Server

Unit: n/a

Format: int

Specifies the number of Receiver connections used towards the SMSC. Also see [Attribute: BindType](#). Relevant when bind type is Transmitter and Receiver.

## Attribute: NumberTransceiverConnections

Scope: Server

Unit: n/a

Format: int

Specifies the number of Transceiver connections used towards the SMSC. Also see [Attribute: BindType](#). Relevant when bind type is Transceiver.

## Attribute: NumberTransmitterConnections

Scope: Server

Unit: n/a

Format: int

Specifies the number of Transmitter connections used towards the SMSC. Also see [Attribute: BindType](#). Relevant when bind type is Transmitter and Receiver.

## 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 on which the plug-in instance executes. This allows for specification of the NIC on which the receiver should bind.

The 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 instance 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>
```

The 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: ReceiveSegmentsWaitTime

Scope: Server

Unit: seconds

Format: int

Specifies the maximum time to wait for the arrival of the segments of a concatenated short message from the SMSC since the arrival of the first segment.

## Attribute: SegmentsLimit

Scope: Server

Unit: n/a

Format: int

Specifies the maximum number of SMPP segments an application is allowed to send when using the Extended Web Services Binary SMS interface.

## **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 service 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 service is restarted or [Operation: resetSMPPConnection](#) is performed.

## Attribute: TransceiverProxyLocalAddressAndPort

Scope: Server

Unit: n/a

Format: String

The address and port which the transceiver of the SMPP SMSC should bind to on the server where the plug-in instance executes. This allows for specifying which NIC the transceiver should bind on.

The 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 instance 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>
```

The 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: 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 instance executes. This allows for specifying which NIC the transmitter should bind on.

The 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 instance 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>
```

The 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.

## Attribute: WindowSize

Scope: Server

Unit: n/a

Format: int [-1 | 1..n]

Specifies the maximum allowed number of unacknowledged SMPP operations between a plug-in instance and an SMSC, enforced for each connection.

This setting applies only to requests sent from the plug-in to the SMSC, not to requests from the SMSC to the plug-in.

A value of -1 indicates that the number of unacknowledged operations is not restricted. Other valid values must be greater than 0 (zero).

## Attribute: WindowWaitTimeout

Scope: Server

Unit: milliseconds

Format: int

Specifies the time to wait before a connection becomes available under the restriction defined in [Attribute: WindowSize](#). Only valid when window size is enforced.

## Operation: addOriginatingAddressTypeMapping

Scope: Cluster

If a tunnelled parameter, `com.bea.wlcp.wlmg.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 19-1 addOriginatingAddressTypeMapping**

<b>addOriginatingAddressTypeMapping</b>	
<b>Parameter</b>	<b>Description</b>
type	Specifies the originating address type to be mapped.
ton	Specifies the ESME Type Of Number (TON). Use: <ul style="list-style-type: none"> <li>• <b>0</b> for UNKNOWN</li> <li>• <b>1</b> for INTERNATIONAL</li> <li>• <b>2</b> for NATIONAL</li> <li>• <b>3</b> for NETWORK</li> <li>• <b>4</b> for SUBSCRIBER</li> <li>• <b>5</b> for ALPHANUMERIC</li> <li>• <b>6</b> for ABBREVIATED</li> </ul>
npi	Specifies the ESME Numbering Plan Indicator (NPI). Use: <ul style="list-style-type: none"> <li>• <b>0</b> for Unknown</li> <li>• <b>1</b> for ISDN (E163/E164)</li> <li>• <b>3</b> for Data (X.121)</li> <li>• <b>4</b> for Telex (F.69)</li> <li>• <b>6</b> for Land Mobile (E.212)</li> <li>• <b>8</b> for National</li> <li>• <b>9</b> for Private</li> <li>• <b>10</b> for ERMES</li> <li>• <b>14</b> for Internet (IP)</li> <li>• <b>18</b> for WAP Client ID</li> </ul>

## Operation: removeOriginatingAddressTypeMapping

Scope: Cluster

Removes an existing TON/NPI address type mapping for a given originating address type.

Signature:

```
removeOriginatingAddressTypeMapping(type: String)
```

**Table 19-2 removeOnlineNotificationInfo**

removeOriginatingAddressTypeMapping	
Parameter	Description
type	Originating address type for the mapping, see <a href="#">Operation: addOriginatingAddressTypeMapping</a> and <a href="#">Operation: listOriginatingAddressTypeMappings</a> .

## 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 19-3 countOfflineNotificationCache**

countOfflineNotificationCache	
Parameter	Description
-	-

## Operation: countOnlineNotificationCache

Read only.

Scope: Cluster

Displays the number of entries in the on-line notification cache.

Signature:

```
countOnlineNotificationCache()
```

**Table 19-4 countOnlineNotificationCache**

countOnlineNotificationCache	
Parameter	Description
-	-

## Operation: 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:

```
countSmsCache()
```

**Table 19-5 countSmsCache**

countSmsCache	
Parameter	Description
-	-

## Operation: enableReceiveSms

Read only.

Scope: Cluster

Adds an offline notification for applications that poll for mobile originated short messages. Those mobile originating Short Messages which match 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,  
appInstanceID: String)
```

**Table 19-6 countSmsCache**

countSmsCache	
Parameter	Description
smsServiceActivationNumber	Destination address of the short message. Prefixed with the URI, for example tel:
criteria	Text to match against to determine if the application should receive the notification.
appInstanceID	ID of the application instance.

## Operation: getOfflineNotificationInfo

Read only.

Scope: Cluster

Displays information about a notification registered off-line, see [Operation: enableReceiveSms](#).

Signature:

```
getOfflineNotificationInfo(correlator: String)
```

**Table 19-7 getOfflineNotificationInfo**

<b>getOfflineNotificationInfo</b>	
<b>Parameter</b>	<b>Description</b>
correlator	Correlator identifying the notification.

## Operation: getOnlineNotificationInfo

Read only.

Scope: Cluster

Displays information about a notification registered by an application

Signature:

```
getOnlineNotificationInfo(correlator: String)
```

**Table 19-8 getOnlineNotificationInfo**

<b>getOnlineNotificationInfo</b>	
<b>Parameter</b>	<b>Description</b>
correlator	Correlator identifying the notification.

## Operation: listOfflineNotificationInfo

Read only.

Scope: Cluster

Displays a list of all notifications registered off-line.

Signature:

```
listOfflineNotificationInfo()
```

**Table 19-9 listOfflineNotificationInfo**

<b>listOfflineNotificationInfo</b>	
<b>Parameter</b>	<b>Description</b>
-	-

## **Operation: listOnlineNotificationInfo**

Read only.

Scope: Cluster

Displays a list of all notifications registered by an application.

Signature:

```
listOnlineNotificationInfo()
```

**Table 19-10 listOnlineNotificationInfo**

<b>listOnlineNotificationInfo</b>	
<b>Parameter</b>	<b>Description</b>
-	-

## **Operation: listOriginatingAddressTypeMappings**

Read only.

Scope: Cluster

Displays a list of all originating address type mappings.

Signature:

```
listOriginatingAddressTypeMappings()
```

**Table 19-11 listOriginatingAddressTypeMappings**

<b>listOriginatingAddressTypeMappings</b>	
<b>Parameter</b>	<b>Description</b>
-	-

## Operation: removeOfflineNotificationInfo

Scope: Cluster

Removes a notification registered off-line.

Signature:

```
removeOfflineNotificationInfo(correlator: String)
```

**Table 19-12 removeOfflineNotificationInfo**

<b>removeOfflineNotificationInfo</b>	
<b>Parameter</b>	<b>Description</b>
correlator	Correlator identifying the notification.

## Operation: removeOnlineNotificationInfo

Scope: Cluster

Removes a notification registered by an application.

Signature:

```
removeOnlineNotificationInfo(correlator: String)
```

**Table 19-13** removeOnlineNotificationInfo

removeOnlineNotificationInfo	
Parameter	Description
correlator	Correlator identifying the notification.

## Operation: resetSMPPConnection

Scope: Server

Resets the connection to the SMSC. If the plug-in instance has stopped trying to connect to the SMSC, invoking this method will restart the reconnect procedure.

Signature:

```
resetSMPPConnection()
```

**Table 19-14** resetSMPPConnection

resetSMPPConnection	
Parameter	Description
-	-

## Operation: startSmsNotification

Scope: Cluster

Registers a notification for mobile originating Short Messages on behalf of an application. Has the same result as if the application used `startSmsNotification` in the Parlay X 2.1 interface `SmsNotificationManager`.

Signature:

```
startSmsNotification(endpoint: String, smsServiceActivationNumber: String,
criteria: String, appInstanceID: String)
```



**Table 19-15 startSmsNotification**

<b>startSmsNotification</b>	
<b>Parameter</b>	<b>Description</b>
endpoint	Notification endpoint implemented by the application. This endpoint implements the Parlay X 2.1 Interface: SmsNotification. Format: URL.
smsServiceActivationNumber	Destination address to the short message address.
criteria	Text in the payload of the Short Message to match to determine the application to receive the notification.
appInstanceID	ID of the application instance.

## Managing and Configuring Short Messaging Communication Services

# Managing and Configuring Multimedia Messaging Communication Services

The following section describes configuration and maintenance attributes and operations for the communication services that expose Parlay X 2.1 Multimedia Messaging Web Services interfaces. The sections also provide a workflow for the configuration:

- [Parlay X 2.1 MultiMedia Messaging/MM7](#)
  - [Properties for Parlay X 2.1 MultiMedia Messaging/MM7](#)
  - [Configuration Workflow for Parlay X 2.1 MultiMedia Messaging/MM7](#)
  - [Provisioning Workflow for Parlay X 2.1 MultiMedia Messaging/MM7](#)
  - [Reference: Attributes and Operations for Parlay X 2.1 MultiMedia Messaging/MM7](#)

## Parlay X 2.1 MultiMedia Messaging/MM7

This section contains a description of the configuration attributes and operations available for Parlay X 2.1 MultiMedia Messaging/MM7 network protocol plug-in instances.

To see a	Refer to
Detailed list of necessary for managing and configuring a plug-in instance	<a href="#">Properties for Parlay X 2.1 MultiMedia Messaging/MM7</a>
Configuration workflow	<a href="#">Configuration Workflow for Parlay X 2.1 MultiMedia Messaging/MM7</a>

To see a	Refer to
List of operations and attributes related to management and provisioning	<a href="#">Provisioning Workflow for Parlay X 2.1 MultiMedia Messaging/MM7</a>
Reference of management attributes and operations	<a href="#">Reference: Attributes and Operations for Parlay X 2.1 MultiMedia Messaging/MM7</a>

## Properties for Parlay X 2.1 MultiMedia Messaging/MM7

Property	Description
Managed object in Administration Console	<domain name>->WLNG-><server name>->Communication Services-><plug-in instance ID>
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName is same as plug-in instance ID Type=om.bea.wlcp.wlng.plugin.multimediamessaging.mm7.management.MessagingManagementMBean
Network protocol plug-in service ID	Plugin_px21_multimedia_messaging_mm7
Network protocol plug-in instance ID	The ID is given when the plug-in instance is created, see <a href="#">Managing and Configuring the Plug-in Manager</a> .
Supported Address Scheme	tel mailto short
North interfaces	com.bea.wlcp.wlng.px21.plugin.MessageNotificationManagerPlugin com.bea.wlcp.wlng.px21.plugin.ReceiveMessagePlugin com.bea.wlcp.wlng.px21.plugin.SendMessagePlugin com.bea.wlcp.wlng.px21.callback.MessageNotificationCallback
Service type	MultimediaMessaging

Property	Description
Exposes to the service communication layer a Java representation of:	Parlay X 3.0 Part 5: Multimedia Messaging
Interfaces with the network nodes using:	MM7
Deployment artifacts	Plugin_px21_multimedia_messaging_mm7.jar, px21_multimedia_messaging_service.jar and multimedia_messaging_mm7.war, packaged in wlng_nt_multimedia_messaging_px21.ear  px21_multimedia_messaging_callback.jar and px21_multimedia_messaging.war, packaged in wlng_at_multimedia_messaging_px21.ear

## Configuration Workflow for Parlay X 2.1 MultiMedia Messaging/MM7

Below is an outline for configuring the plug-in using the Network Gatekeeper Management Console:

1. Create one or more instances of the plug-in service: see [Managing and Configuring the Plug-in Manager](#). Use the plug-in service ID as detailed in [Properties for Parlay X 2.1 MultiMedia Messaging/MM7](#).
2. Using the Management Console or an MBean browser, select the MBean for the plug-in instance. The MBean display name is the same as the plug-in instance ID given when the plug-in instance was created.
3. Configure the behavior of the plug-in instance:
  - [Attribute: HTTPBasicAuthentication](#): if using HTTP basic authentication also define:
    - [Attribute: HTTPBasicAuthenticationUsername](#)
    - [Attribute: HTTPBasicAuthenticationPassword](#)
  - [Attribute: DefaultPriority](#)
  - [Attribute: MM7Version](#)
  - [Attribute: Mm7relayserverAddress](#)

- [Attribute: VaspId](#)
  - [Attribute: VasId](#)
  - [Attribute: RequestDeliveryReportFlag](#)
  - [Attribute: XSDVersion](#)
4. Specify heartbeat behavior, see [Configuring Heartbeats](#).
  5. Set up the routing rules to the plug-in instance; see [Configuring the Plug-in Manager](#). Use the plug-in instance ID and address schemes detailed in [Properties for Parlay X 2.1 MultiMedia Messaging/MM7](#).
  6. Provide the administrator of the MM7 server with the URL to which the MM7 server should deliver mobile-originated messages and delivery reports. The default URL is  
`http://<WLANG NT IP Address>:<port>/<server>:<port>/<context-root>/<Plug-in instance ID>`  
default <context-root> is mmm-mm7
  7. If desired, create and load a node SLA, see:  
[Defining Global Node and Service Provider Group Node SLAs](#)  
[Managing Application SLAs](#)

Continue with the provisioning of service provider accounts and application accounts.

## Provisioning Workflow for Parlay X 2.1 MultiMedia Messaging/MM7

1. To register offline notifications, that is, to specify that mobile originated messages should not result in notifications to an application, but instead be stored in Network Gatekeeper for polling, use [Operation: enableReceiveMms](#). Use the following operations to manage the offline registrations:
  - [Operation: listOfflineNotificationInfo](#)
  - [Operation: getOfflineNotificationInfo](#)
  - [Operation: removeOfflineNotificationInfo](#)
2. To register online notifications, that is, to manage registrations for mobile originated messages on behalf of an application, use: [Operation: startMessageNotification](#). Use the following operations to manage the online registrations:

- Operation: [listOnlineNotificationInfo](#)
- Operation: [getOnlineNotificationInfo](#)
- Operation: [removeOnlineNotificationInfo](#)

## Reference: Attributes and Operations for Parlay X 2.1 MultiMedia Messaging/MM7

Managed object: Communication Services-><plug-in instance ID>

MBean:

com.bea.wlcp.wlmg.plugin.multimediamessaging.mm7.management.MessagingManagementMBean

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

- Attribute: [HTTPBasicAuthentication](#)
- Attribute: [HTTPBasicAuthenticationUsername](#)
- Attribute: [HTTPBasicAuthenticationPassword](#)
- Attribute: [DefaultPriority](#)
- Attribute: [Mm7relayserverAddress](#)
- Attribute: [MM7Version](#)
- Attribute: [RequestDeliveryReportFlag](#)
- Attribute: [VaspId](#)
- Attribute: [VasId](#)
- Attribute: [XSDVersion](#)
- Operation: [enableReceiveMms](#)
- Operation: [getOfflineNotificationInfo](#)
- Operation: [getOnlineNotificationInfo](#)
- Operation: [listOfflineNotificationInfo](#)
- Operation: [listOnlineNotificationInfo](#)

- [Operation: removeOfflineNotificationInfo](#)
- [Operation: removeOnlineNotificationInfo](#)
- [Operation: startMessageNotification](#)

## Attribute: HTTPBasicAuthentication

Scope: Cluster

Unit: n/a

Format: Boolean

Specifies if HTTP basic authentication shall be used for authentication with the MM7 server.

Set to `true` if HTTP basic authentication shall be used, otherwise `false`.

If `true`, [Attribute: HTTPBasicAuthenticationUsername](#) and [Attribute: HTTPBasicAuthenticationPassword](#) must be specified.

## Attribute: HTTPBasicAuthenticationUsername

Scope: Cluster

Unit: n/a

Format: String

The username to use for HTTP basic authentication towards the MM7 server.

## Attribute: HTTPBasicAuthenticationPassword

Scope: Cluster

Unit: n/a

Format: String

The password to use for HTTP basic authentication towards the MM7 server.

## Attribute: DefaultPriority

Scope: Cluster

Unit: n/a

Format: String



Specifies the default priority for sent MMSes. Enter one the following:

- normal
- high
- low

## **Attribute: MM7Version**

Scope: Cluster

Unit: n/a

Format: String

Specifies the version of the MM7 protocol to be used. Applicable versions are:

- 5.3.0

## **Attribute: Mm7relayserverAddress**

Scope: Cluster

Unit: n/a

Format: String

Specifies the address to the MM7 Relay Server. The address is an HTTP URL.

## **Attribute: VaspId**

Scope: Cluster

Unit: n/a

Format: String

Specifies the VASP (Value Added Service Provider) ID to be used for the plug-in instance when connecting to the MMSC.

## **Attribute: VasId**

Scope: Cluster

Unit: n/a

Format: String

Specifies the VAS (Value Added Service) ID to be used for the plug-in instance when connecting to the MMSC.

## Attribute: RequestDeliveryReportFlag

Scope: Cluster

Unit: n/a

Format: Integer

Specifies how the plug-in instance requests and handles delivery reports for sent messages. Enter one of the following:

- 0: Delivery notifications are not processed, which means that no polling functionality is available to the applications using the communication service.
- 1: Delivery notifications are processed if the application provided a receiptRequest in the SendMessage requests or the application provided a tunnelled parameter with ID `com.bea.wlcp.wlng.plugin.multimediamessaging.RequestDeliveryReportFlag` with the value `true` in the SOAP header of the SendMessage request.
- 2: Delivery notifications are always processed.

## Attribute: XSDVersion

Scope: Server

Unit: n/a

Format: String [REL-5-MM7-1-0, REL-5-MM7-1-2]

The MM7 xsd version that should be used for requests towards the MMSC.

Enter one of the following:

- REL-5-MM7-1-0 to use an altered version of the REL-5-MM7-1-0.xsd. The altered version allows use of delivery notifications when the MMC-S requires this version of the xsd. This is a requirement when connecting to, among others, Comverse MMSCs.
- REL-5-MM7-1-2, to use REL-5-MM7-1-2.xsd.

## Operation: enableReceiveMms

Scope: Cluster

Adds an offline notification for applications that will poll for mobile originated messages. Mobile originated messages matching this notification will not result in a callback to an application. Instead the application has to use the correlator returned by this method and poll for new messages.

Returns the correlator uniquely identifying the new notification.

Signature:

```
enableReceiveMms(shortcode: String, criteria: String, appInstanceID:
String)
```

**Table 20-1 enableReceiveMms**

enableReceiveMms	
Parameter	Description
shortcode	The destination address or service activation number of the Multimedia message. Prefixed with the URI, for example tel:
criteria	The first word in the text in the subject field of the MMS to match. Exact matches only.
appInstanceID	The application instance ID associated with the notification.

## Operation: getOfflineNotificationInfo

Scope: Cluster

Displays information about a notification registered offline, see [Operation: enableReceiveMms](#).

Signature:

```
getOfflineNotificationInfo(correlator: String)
```

**Table 20-2** `getOfflineNotificationInfo`

<code>getOfflineNotificationInfo</code>	
Parameter	Description
correlator	Correlator identifying the notification.

## Operation: `getOnlineNotificationInfo`

Scope: Cluster

Displays information about a notification registered by an application or via [Operation: `startMessageNotification`](#).

Signature:

```
getOnlineNotificationInfo(correlator: String)
```

**Table 20-3** `getOfflineNotificationInfo`

<code>getOfflineNotificationInfo</code>	
Parameter	Description
correlator	Correlator identifying the notification.

## Operation: `listOfflineNotificationInfo`

Scope: Cluster

Displays a list of all notifications registered offline using [Operation: `enableReceiveMms`](#).

Signature:

```
listOfflineNotificationInfo()
```

**Table 20-4 listOfflineNotificationInfo**

listOfflineNotificationInfo	
Parameter	Description
-	-

## Operation: listOnlineNotificationInfo

Scope: Cluster

Displays a list of all notifications registered online by the application, or via [Operation: startMessageNotification](#)

Signature:

```
listOnlineNotificationInfo()
```

**Table 20-5 listOnlineNotificationInfo**

listOnlineNotificationInfo	
Parameter	Description
-	-

## Operation: removeOfflineNotificationInfo

Scope: Cluster

Removes a notification registered offline using [Operation: enableReceiveMms](#).

Signature:

```
removeOfflineNotificationInfo(Registration Identifier: String)
```

**Table 20-6 removeOfflineNotificationInfo**

<b>removeOfflineNotificationInfo</b>	
<b>Parameter</b>	<b>Description</b>
Registration Identifier	ID of the notification.

## Operation: removeOnlineNotificationInfo

Scope: Cluster

Removes a notification registered by an application or on behalf of an application via [Operation: startMessageNotification](#).

Signature:

```
removeOnlineNotificationInfo(Registration Identifier: String)
```

**Table 20-7 removeOnlineNotificationInfo**

<b>removeOnlineNotificationInfo</b>	
<b>Parameter</b>	<b>Description</b>
Correlator	ID of the notification.

## Operation: startMessageNotification

Scope: Cluster

Creates an online notification on behalf of an application. Produces the same results as if an application registered for notifications using the method `startMessageNotification` in Parlay X 2.1 Multimedia Messaging interface `MessageNotificationManager`.

This operation can be used, for example, if the application is not allowed to register for notifications by restrictions defined in its SLA. Returns a correlator that uniquely identifies the notification.

Signature:

```
startMessageNotification(endpoint: String, shortcode: String, criteria:
String, appInstanceID: String)
```

**Table 20-8 startMessageNotification**

<b>startMessageNotification</b>	
<b>Parameter</b>	<b>Description</b>
endpoint	Notification endpoint implemented by the application. This endpoint implements the Parlay X 2.1 Multimedia Messaging interface MessageNotification. Format: URL.
shortcode	Destination address for the MMS. Must have the suffix “tel:”, for example tel:1234
criteria	The first word in the text in the subject field of the MMS to match. Exact matches only.
appInstanceID	ID of application instance for the application.

## Managing and Configuring Multimedia Messaging Communication Services



# Managing and Configuring Terminal Location Communication Services

The following section describes configuration and maintenance attributes and operations for the communication services that expose Parlay X 2.1 Terminal Location Web Services interfaces. The sections also provide a workflow for the configuration:

- [Parlay X 2.1 Terminal Location/MLP](#)
  - [Configuration Workflow for Parlay X 2.1 Terminal Location/MLP](#)
  - [Provisioning Workflow for Parlay X 2.1 Terminal Location/MLP](#)
  - [Management Operations for Parlay X 2.1 Terminal Location/MLP](#)
  - [Reference: Attributes and Operations for Parlay X 2.1 Terminal Location/MLP](#)

## Parlay X 2.1 Terminal Location/MLP

This section contains a description of the configuration attributes and operations available for Parlay X 2.1 Terminal Location/MLP network protocol plug-in instance.

To see a	Refer to
Detailed list of necessary for managing and configuring the plug-in instance	<a href="#">Properties for Parlay X 2.1 Terminal Location/MLP</a>
Configuration workflow	<a href="#">Configuration Workflow for Parlay X 2.1 Terminal Location/MLP</a>

To see a	Refer to
List of operations and attributes related to management and provisioning	<a href="#">Provisioning Workflow for Parlay X 2.1 Terminal Location/MLP</a> <a href="#">Management Operations for Parlay X 2.1 Terminal Location/MLP</a>
Reference of management attributes and operations	<a href="#">Reference: Attributes and Operations for Parlay X 2.1 Terminal Location/MLP</a>

## Properties for Parlay X 2.1 Terminal Location/MLP

Property	Description
Managed object in Administration Console	<domain name>->WLNG-><server name>->Communication Services-><plug-in instance ID>
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName is same as plug-in instance ID Type=com.bea.wlcp.wlng.plugin.terminallocation.mlp.management.TerminalLocationMLPMBean
Network protocol plug-in service ID	Plugin_px21_terminal_location_mlp
Network protocol plug-in instance ID	The ID is given when the plug-in instance is created, see <a href="#">Managing and Configuring the Plug-in Manager</a> .
Supported Address Scheme	tel
North interfaces	com.bea.wlcp.wlng.px21.plugin.TerminalLocationPlugin com.bea.wlcp.wlng.px21.plugin.TerminalLocationNotificationManagerPlugin com.bea.wlcp.wlng.px21.callback.TerminalLocationNotificationCallback
Service type	TerminalLocation

Property	Description
Exposes to the service communication layer a Java representation of:	Parlay X 2.1 Part 9: Terminal Location
Interfaces with the network nodes using:	MLP 3.0/3.2
Deployment artifacts	Plugin_px21_terminal_location_mlp.jar, px21_terminal_location_service.jar, and terminal_location_mlp.war, packaged in wlng_nt_terminal_location_px21.ear  px21_terminal_location.war, and px21_terminal_location_callback.jar packaged in wlng_at_terminal_location_px21.ear

## Configuration Workflow for Parlay X 2.1 Terminal Location/MLP

Below is an outline for configuring the plug-in instance using the Network Gatekeeper Administration Console:

1. Create one or more instances of the plug-in service: see [Managing and Configuring the Plug-in Manager](#). Use the plug-in service ID as detailed in [Properties for Parlay X 2.1 Terminal Location/MLP](#).
2. Using the Management Console or an MBean browser, select the MBean for the plug-in instance. The MBean display name is the same as the plug-in instance ID given when the plug-in instance was created.
3. Configure the attributes of the plug-in instance:
  - Attribute: [CharacterEncoding](#)
  - Attribute: [CleanupInterval](#)
  - Attribute: [MaxDuration](#)
  - Attribute: [MlpAltitudeSupported](#)
  - Attribute: [MlpLocationEstimates](#)
  - Attribute: [MlpPushAddr](#)
  - Attribute: [MlpRequestType](#)

- [Attribute: MlpServerUrl](#)
  - [Attribute: MlpSrsName](#)
  - [Attribute: MlpVersionSupported](#)
  - [Attribute: Password](#)
  - [Attribute: Requestor](#)
  - [Attribute: RequestTimeout](#)
  - [Attribute: ServiceId](#)
  - [Attribute: Username](#)
  - [Attribute: XMLDoctypeTagUsage](#).
4. Specify heartbeat behavior, see [Configuring Heartbeats](#).
  5. Set up the routing rules to the plug-in instance; see [Configuring the Plug-in Manager](#). Use the plug-in instance ID and address schemes detailed in [Properties for Parlay X 2.1 Terminal Location/MLP](#).
  6. If desired, create and load a node SLA, see:
    - [Defining Global Node and Service Provider Group Node SLAs](#)
    - [Managing Application SLAs](#)

Continue with the provisioning of service provider accounts and application accounts.

## Provisioning Workflow for Parlay X 2.1 Terminal Location/MLP

No provisioning operations.

## Management Operations for Parlay X 2.1 Terminal Location/MLP

No management operations.

## Reference: Attributes and Operations for Parlay X 2.1 Terminal Location/MLP

Below is a list of attributes for configuration and maintenance:

- Attribute: CharacterEncoding
- Attribute: CleanupInterval
- Attribute: MaxDuration
- Attribute: MlpAltitudeSupported
- Attribute: MlpLocationEstimates
- Attribute: MlpPushAddr
- Attribute: MlpRequestType
- Attribute: MlpServerUrl
- Attribute: MlpSrsName
- Attribute: MlpVersionSupported
- Attribute: Password
- Attribute: Requestor
- Attribute: RequestTimeout
- Attribute: ServiceId
- Attribute: Username
- Attribute: XMLDoctypeTagUsage

## **Attribute: MlpServerUrl**

Scope: Cluster

Unit: n/a

Format: URL

Specifies the MLP server's URL.

## **Attribute: Username**

Scope: Cluster

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: Password

Scope: Cluster

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: ServiceId

Scope: Cluster

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: Requestor

Scope: Cluster

Unit: n/a

Format: String

Specifies the 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: MlpRequestType

Scope: Cluster

Unit: n/a

Format: String [eme\_lir | slir]

Specifies which type of location request to use towards the MLP server.

Valid values are:

- eme\_lir for EME\_LIR (Emergency location request)
- slir for SLIR (Standard location request)

Defines the DTD to be used for constructing the request towards the MLP server.

## Attribute: MlpPushAddr

Scope: Server

Unit: n/a

Format: URL

Specifies the callback URL to which the MLP server delivers location reports, periodic or triggered, which is the URL at which the plug-in instance listens for location reports. The format for the URL is:

`http://<ipaddressOfNTMachine>:<portOfWLS>/tl-mlp/mlp_client`

For example, `http://172.16.0.0:8001/tl-mlp/mlp_client`

## Attribute: CharacterEncoding

Scope: Cluster

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: MlpSrsName

Scope: Cluster

Unit: n/a

Format: String

Specifies requested MLP srsName attribute.

Normally this is `www.epsg.org#4326`

## Attribute: MlpVersionSupported

Scope: Cluster

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: MlpLocationEstimates

Scope: Cluster

Unit: n/a

Format: Boolean [true | false], not case sensitive, all strings other than true are treated as false.

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: MlpAltitudeSupported

Scope: Cluster

Unit: n/a

Format: Boolean [true | false], not case sensitive, all strings other than true are treated as false.

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 instance operates in MLP 3.2 mode, see [Attribute: MlpVersionSupported](#).

## Attribute: XMLDoctypeTagUsage

Scope: Cluster

Unit: n/a



Format: Boolean [true | false], not case sensitive, all strings other than true are treated as 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

## Attribute: RequestTimeout

Scope: Cluster

Unit: seconds

Format: int [0-3600]

Specifies the HTTP time-out for MLP requests.

## Attribute: CleanupInterval

Scope: Cluster

Unit: seconds

Format: int [0-3600]

Specifies the time interval at which periodic notification expiration checks are performed.

## Attribute: MaxDuration

Scope: Cluster

Unit: seconds

Format: int

Specifies the maximum duration for a periodic location request.

Rejects `startPeriodicNotification` and `startGeographicalNotification` requests on the `TerminalLocationNotificationManager` interface if the duration is larger than this value.

If the duration is not provided in the request, this value is used.

## Managing and Configuring Terminal Location Communication Services

# Managing and Configuring Audio Call Communication Services

The following section describes configuration and maintenance attributes and operations for the communication services that expose Parlay 3.0 Audio Call Web Services interfaces. The sections also provide a workflow for the configuration:

- [Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction](#)
  - [Configuration Workflow for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction](#)
  - [Reference: Attributes and Operations for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction](#)

## Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction

This section contains a description of the configuration attributes and operations available for the Parlay X 3.0 Audio Call /Parlay MultiParty Call Control and Call User Interaction network protocol plug-in instance.

Most of the configuration is done in the OSA Access module, but with configuration parameters for Parlay MultiParty Call Control. See [Managing OSA/Parlay Gateway Connections using Parlay\\_Access](#)

For a configuration workflow, see [Configuration Workflow for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction](#).

To see a	Refer to
Detailed list of necessary for managing and configuring the plug-in instance	<a href="#">Properties for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction</a>
Configuration workflow	<a href="#">Configuration Workflow for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction</a>
List of operations and attributes related to management and provisioning	None.
Reference of management attributes and operations	<a href="#">Reference: Attributes and Operations for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction</a>

## Properties for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction

Property	Description
Managed object in Administration Console	<domain name>->WLNG-><server name>->Communication Services->Plugin_px30_audio_call_parlay_mpcc_cui
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName=Plugin_px30_audio_call_parlay_mpcc_cui Type=com.bea.wlcp.wlng.plugin.ac.parlay.management.AudioCallManagementMBean
Network protocol plug-in service ID	Plugin_px30_audio_call_parlay_mpcc_cui

Property	Description
Network protocol plug-in instance ID	Plugin_px30_audio_call_parlay_mpcc_cui
Supported Address Scheme	tel:
North interfaces	com.bea.wlcp.wlng.px30.plugin.AudioCallPlayMediaPlugin com.bea.wlcp.wlng.px30.plugin.AudioCallCaptureMediaPlugin
Service type	AudioCall
Exposes to the service communication layer a Java representation of:	Parlay X 3.0 Part 11: Audio Call
Interfaces with the network nodes using:	Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control SCF; Subpart 7: MultiParty Call Control Service Open Service Access (OSA); Application Programming Interface (API); Part 5: User Interaction SCF
Deployment artifacts	osa_access.jar, Plugin_px30_audio_call_parlay_mpcc_cui.jar, and px30_audio_call_service.jar, packaged in wlng_nt_audio_call_px30.ear px30_audio_call.war, packaged in wlng_at_audio_call_px30.ear

This plug-in service does not support multiple instantiation using the Plug-in Manager. There is a one to one mapping between plug-in service and plug-in instance. The plug-in instance is created when the plug-in service is started.

## Configuration Workflow for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction

Below is an outline for configuring the plug-in using Network Gatekeeper Management Console:

1. Using the Management Console or an MBean browser, select the MBean detailed in [Properties for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction](#).
2. Configure the behavior of the plug-in instance:
  - [Attribute: Retentiontime](#)

- Attribute: CollectStartTimeout
  - Attribute: CollectInterCharTimeout
  - Attribute: ShutdownTimerInterval
  - Attribute: Language
  - Attribute: ChargingAllowed
  - Attribute: Retentiontime
  - Attribute: RepeatIndicator
  - Attribute: ResponseRequested
  - Attribute: MaxDigits
  - Attribute: MinDigits
  - Attribute: EndSequence
  - Attribute: CollectStartTimeout
  - Attribute: CollectInterCharTimeout
3. Gather information about the OSA Gateway and configure the MultiParty Call Control part of the protocol translator accordingly. The following information needs to be obtained from the OSA Gateway administrator and configured in the Parlay\_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.
  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 Managing OSA/Parlay Gateway Connections using Parlay\\_Access](#) for the MultiParty Call Control part of the plug-in instance.

5. Gather information about the OSA Gateway and configure the Call User Interaction part of the protocol translator 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.
6. Set up 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 [Managing OSA/Parlay Gateway Connections using Parlay\\_Access](#) for the Generic User Interaction part of the plug-in instance.
7. If desired, create and load a node SLA, see:

[Defining Global Node and Service Provider Group Node SLAs](#)

[Managing Application SLAs](#)

**Note:** It is not necessary to set up routing rules to the plug-in instance.

## Reference: Attributes and Operations for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction

Below is a list of attributes for configuration and maintenance:

- [Attribute: CollectStartTimeout](#)
- [Attribute: CollectInterCharTimeout](#)
- [Attribute: Language](#)
- [Attribute: ChargingAllowed](#)

- [Attribute: Retentiontime](#)
- [Attribute: RepeatIndicator](#)
- [Attribute: ResponseRequested](#)
- [Attribute: RetensionTime](#)
- [Attribute: ShutdownTimerInterval](#)
- [Attribute: MaxDigits](#)
- [Attribute: MinDigits](#)
- [Attribute: EndSequence](#)

## Attribute: CollectStartTimeout

Scope: Cluster

Unit: Seconds

Specifies the start time-out when collecting user input, the value for the first character time-out timer.

The value corresponds to the parameter `TpUICollectCriteria.StartTimeout` in the `sendInfoAndCollectReq` requests towards the Generic User Interaction SCS.

## Attribute: CollectInterCharTimeout

Scope: Cluster

Unit: Seconds

Specifies the inter-character time-out when collecting user input, the value for the inter-character time-out timer.

The value corresponds to the parameter `TpUICollectCriteria.InterCharTimeOut` in the `sendInfoAndCollectReq` requests towards the Generic User Interaction SCS.

## Attribute: Language

Scope: Cluster

Unit: n/a

Format: String according to valid language strings as defined in ISO 639.



Specifies the language of the message to be played for the call participant.

## Attribute: ChargingAllowed

Scope: Cluster

Unit: n/a

Format: Boolean

Specifies whether charging is allowed.

- `true` if an application is allowed to specify charging information when playing a message (Parlay X operation `playAudioMessage`).
- `False` if not.

## Attribute: Retentiontime

Scope: Cluster

Unit: Seconds

Format: int [0-]

Specifies the length of time status information about a message (Parlay X `MessageStatus`) is stored after play operation has finished or an error related to the playing of a message has been received.

## Attribute: RepeatIndicator

Scope: Cluster

Unit: n/a

Format: int

Specifies the number of times a message should be played to the call participant.

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 [1,2,4]

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

## Attribute: RetentionTime

Scope: Cluster

Unit: Seconds

Format: int

Specifies the time-interval for which status information is retained after a message is played or an error occurs.

## Attribute: ShutdownTimerInterval

Scope: Server

Unit: Seconds

Format: int

Specifies the time-interval to wait for call sessions to end before terminating when performing a graceful shutdown.

## Attribute: MaxDigits

Scope: Server

Unit: n/a

Format: int [1 -]

Specifies the maximum number of digits that can be collected from the call participant.

The value corresponds to the parameter `TpUICollectCriteria.MaxLength` in the `sendInfoAndCollectReq` requests towards the Generic User Interaction SCS.

## Attribute: MinDigits

Scope: Server

Unit: n/a

Format: int [1 -]

Specifies the minimum number of digits that can be collected from the call participant.

The value corresponds to the parameter `TpUICollectCriteria.MinLength` in the `sendInfoAndCollectReq` requests towards the Generic User Interaction SCS.

## Attribute: EndSequence

Scope: Server

Unit: n/a

Format: String

Specifies the digit to be used for ending collection of data of various lengths from a call participant.

The value corresponds to the parameter `TpUICollectCriteria.EndSequence` in the `sendInfoAndCollectReq` requests towards the Generic User Interaction SCS.

## Managing and Configuring Audio Call Communication Services

# Managing and Configuring the Presence Communication Services

The following section describes configuration and maintenance attributes and operations for the communication services that expose Parlay X 2.1 Presence Web Services interfaces:

- [Parlay X 2.1 Presence/SIP](#)
  - [URI Cache](#)
  - [Subscriptions Cache](#)
  - [Notifications Cache](#)
  - [Configuration Workflow for Parlay X 2.1 Presence/SIP](#)
  - [Provisioning Workflow for Parlay X 2.1 Presence/SIP](#)
  - [Management Operations for Parlay X 2.1 Presence/SIP](#)
  - [Reference: Attributes and Operations for Parlay X 2.1 Presence/SIP](#)

## Parlay X 2.1 Presence/SIP

This section contains a description of the configuration attributes and operations available for the Parlay X 2.1 Presence/SIP network protocol plug-in.instance.

Parlay X 2.1 Presence/SIP uses two parts for SIP connectivity, a part that executes as a network protocol plug-in instance in Network Gatekeeper container, and a part that executes as a SIP application in the SIP Server container.

The plug-in instance uses a set of caches:

- [URI Cache](#)
- [Subscriptions Cache](#)
- [Subscriptions Cache](#)

To see a	Refer to
Detailed list of necessary for managing and configuring the plug-in instance	<a href="#">Properties for Parlay X 2.1 Presence/SIP</a>
Configuration workflow	<a href="#">Configuration Workflow for Parlay X 2.1 Presence/SIP</a>
List of operations and attributes related to management and provisioning	<a href="#">Provisioning Workflow for Parlay X 2.1 Presence/SIP</a> <a href="#">Management Operations for Parlay X 2.1 Presence/SIP</a>
Reference of management attributes and operations	<a href="#">Reference: Attributes and Operations for Parlay X 2.1 Presence/SIP</a>

## 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 application instance ID, that is using the interface must be mapped to a URI. The URI mapping is configured as a part of the service provider and application provisioning workflow and is then 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 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 registered 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.

## Properties for Parlay X 2.1 Presence/SIP

Property	Description
Managed object in Administration Console	<domain name>->WLNG-><server name>->Communication Services->Plugin_presence_sip
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName=Plugin_px21_presence_sip Type=com.bea.wlcp.wlng.plugin.presence.sip.impl.PresenceMBean
Network protocol plug-in service ID	Plugin_px21_presence_sip
Network protocol plug-in instance ID	Plugin_px21_presence_sip
Supported Address Scheme	sip
North interfaces	com.bea.wlcp.wlng.px21.plugin.PresenceConsumerPlugin com.bea.wlcp.wlng.px21.plugin.PresenceSupplierPlugin com.bea.wlcp.wlng.px21.callback.PresenceNotificationCallback
Service type	Presence
Exposes to the service communication layer a Java representation of:	Parlay X 2.1 Part 14: Presence

Property	Description
Interfaces with the network nodes using:	SIP: Session Initiation Protocol, RFC 3261.
Deployment artifacts	px21_presence_service.jar, presence_sip.jar, andplug_in_presence_callback_ejb.jar, packaged in wlng_nt_presence_px21.ear  px21_presencel.war and px21_presense_callback.jar, packaged in wlng_at_presence_px21.ear  wlss_presence_ejb.jar and wlss_presence_web.war, packaged in wlss-int.ear

This plug-in service does not support multiple instantiation using the Plug-in Manager. There is a one to one mapping between plug-in service and plug-in instance. The plug-in instance is created when the plug-in service is started.

## Configuration Workflow for Parlay X 2.1 Presence/SIP

1. Using the Management Console or an MBean browser, select the MBean detailed in [Properties for Parlay X 2.1 Presence/SIP](#)
2. Configure the attributes of the network protocol plug-in instance:
  - [Attribute: DefaultNotificationCount](#)
  - [Attribute: DefaultNotificationDuration](#)
  - [Attribute: NotificationCleanupTimerValue](#)
  - [Attribute: NotificationCleanupTimerValue](#)
3. Configure connection information to the SIP server:
  - [Attribute: SIPNodeUrl](#)
  - [Attribute: SIPNodeUsername](#)
  - [Attribute: SIPNodePassword](#)
4. Configure connection information from the SIP server to Gatekeeper, see [Configuring WebLogic SIP Server Integration](#).



5. Set up the routing rules to the plug-in instance: see [Configuring the Plug-in Manager](#). Use the plug-in instance ID and address schemes detailed in [Properties for Parlay X 2.1 Presence/SIP](#).
6. If desired, create and load a node SLA, see:

[Defining Global Node and Service Provider Group Node SLAs](#)

[Managing Application SLAs](#)

Move on to the provisioning of service provider accounts and application accounts.

## Provisioning Workflow for Parlay X 2.1 Presence/SIP

For every application a mapping between a SIP URI and application instance ID needs to be defined using [Operation: setApplicationInstanceSIPURI](#).

To display the mapping, use [Operation: getApplicationInstanceSIPURI](#).

If an application is deleted, the data for the application needs to be removed using [Operation: removeApplicationInstanceFromCache](#).

## Management Operations for Parlay X 2.1 Presence/SIP

- [Operation: clearCache](#)
- [Operation: connectToWLSS](#)
- [Operation: listNotificationsCache](#)
- [Operation: listSubscriptionsCache](#)
- [Operation: listURIMappingCache](#)

## Reference: Attributes and Operations for Parlay X 2.1 Presence/SIP

Managed object: Communication Services->Plugin\_presence\_sip

MBean: com.bea.wlcp.wlmg.plugin.presence.sip.impl.PresenceMBean

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

- [Attribute: DefaultNotificationCount](#)
- [Attribute: DefaultNotificationDuration](#)

- Attribute: NotificationCleanupTimerValue
- Attribute: SIPNodeUrl
- Attribute: SIPNodeUsername
- Attribute: SIPNodePassword
- Attribute: SubscriptionCleanupTimerValue
- Operation: clearCache
- Operation: getApplicationInstance
- Operation: getApplicationInstanceSIPURI
- Operation: listNotificationsCache
- Operation: listSubscriptionsCache
- Operation: listURImappingCache
- Operation: removeApplicationInstanceFromCache
- Operation: removeNotification
- Operation: removeSubscription
- Operation: setApplicationInstanceSIPURI

## 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

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: SIPNodeUrl

Scope: Cluster

Unit: n/a

Format: String in URL format

Specifies the WebLogic SIP Server JNDI URL. This was configured when WebLogic SIP Server was installed.

WebLogic SIP Server naming service contains the host name (or IP address) and the port number identifying `wlss.jndi.uri`.

Example: `t3://127.0.0.1:7002`

**Note:** This is not the same as the Network Gatekeeper naming service.

## Attribute: SIPNodeUsername

Scope: Cluster

Unit: n/a

Format: String

Specifies the WebLogic SIP Server JNDI access username for the SIP application part of the SIP integration for Third Party Call. This is a regular WebLogic SIP Server user.

## Attribute: SIPNodePassword

Scope: Cluster

Unit: n/a

Format: String

Specifies the password associated with [Attribute: SIPNodeUsername](#).

## 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 subscriptions. If an old subscription is found during the check it will be removed a callback is made to the application.

## Attribute: PresenceServerAddress

Scope: Cluster

Unit: n/a

Format: String formatted as a SIP URI

Specifies 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

## Attribute: SubscribeExpiryValue

Scope: Cluster

Unit: seconds

Format: int

Specifies the maximum lifetime of a subscription.

This value might not be accepted by the Presence Server. The Presence Server may override this expiry value, and give the suggested value to be used in the first NOTIFY sent to the plug-in instance. In that case, the lifetime for the presence subscription will be according to the value received from the Presence Server.

## Operation: clearCache

Scope: Cluster

Clears one or all caches used by this plug-in instance.

**Note:** Use this method with care.

Signature:

```
clearCache(cacheToClear: String)
```

**Table 23-1 clearCache**

clearCache	
Parameter	Description
cacheToClear	Name of the cache to clear. Valid options: <ul style="list-style-type: none"> <li>• NOTIFICATIONS -clears the notification cache</li> <li>• SUBSCRIPTIONS -clears the subscriptions cache</li> <li>• URIMAPPINGS -clears the URI mappings cache</li> <li>• ALL -clears all caches.</li> </ul>

## Operation: connectToWLSS

Scope: Cluster

Connects to the PresenceConsumer EJB in the SIP Server.

Signature:

```
connectToWLSS()
```

**Table 23-2 connectToWLSS**

---

<b>connectToWLSS</b>	
<b>Parameter</b>	<b>Description</b>
-	-

---

## Operation: **getApplicationInstance**

Scope: Cluster

Displays the application instance ID associated with a SIP URI. The application instance is identifies an application.

Signature:

```
getApplicationInstance(Uri: String)
```

**Table 23-3 getApplicationInstance**

---

<b>getApplicationInstanceGroup</b>	
<b>Parameter</b>	<b>Description</b>
Uri	The SIP URI.

---

## Operation: **getApplicationInstanceSIPURI**

Scope: Cluster

Displays the SIP URI associated with an application instance. The application instance is used by an application.

Signature:

```
getApplicationInstanceSIPURI(ApplicationInstanceID: String)
```

**Table 23-4** `getApplicationInstanceSIPURI`

<code>getApplicationInstanceSIPURI</code>	
Parameter	Description
<code>ApplicationInstanceID</code>	ID of the application instance.

## 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:

```
listNotificationsCache()
```

**Table 23-5** `listNotificationsCache`

<code>listNotificationsCache</code>	
Parameter	Description
-	-

## 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 23-6 listSubscriptionsCache**

listSubscriptionsCache	
Parameter	Description
-	-

## 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()
```

**Table 23-7 listURIMappingCache**

listURIMappingCache	
Parameter	Description
-	-

## Operation: removeApplicationInstanceFromCache

Scope: Cluster

Removes entries that are associated with an application instance from the URI mappings cache. If an application instance has been removed, the associated entries in the cache must be removed, too.

Signature:

```
removeApplicationInstanceFromCache(ApplicationInstance: String)
```



**Table 23-8 removeApplicationInstanceFromCache**

<b>removeApplicationInstanceFromCache</b>	
<b>Parameter</b>	<b>Description</b>
ApplicationInstance	ID of the application instance.

## Operation: removeNotification

Scope: Cluster

Removes a notification. The application will not be notified that the notification has been removed.

Signature:

```
removeNotification(ApplicationInstanceID: String, Presentity: String)
```

**Table 23-9 removeNotification**

<b>removeNotification</b>	
<b>Parameter</b>	<b>Description</b>
ApplicationInstanceID	ID of the application instance.
Presentity	ID of the presentity.

## Operation: removeSubscription

Scope: Cluster

Removes a subscription and notifications. The application will not be notified that the subscription has been removed.

Signature:

```
removeSubscription(ApplicationInstanceID: String, Presentity: String)
```

**Table 23-10** removeSubscription

<b>removeSubscription</b>	
<b>Parameter</b>	<b>Description</b>
ApplicationInstanceID	ID of the application instance.
Presentity	ID of the presentity.

## Operation: setApplicationInstanceSIPURI

Scope: Cluster

Associates a SIP URI with an application instance. See [URI Cache](#).

Signature:

```
setApplicationInstanceSIPURI(appInstGrpId: String, URI: String)
```

**Table 23-11** setApplicationInstanceSIPURI

<b>setApplicationInstanceSIPURI</b>	
<b>Parameter</b>	<b>Description</b>
ApplicationInstanceID	ID of the application instance.
URI	SIP URI. For example: sip:name@somedomain.org

# Managing and Configuring Subscriber Profile Communication Services

The following section describes configuration and maintenance attributes and operations for the Extended Web Services Subscriber Profile communication service. It also provides a workflow for the configuration:

- [Extended Web Services Subscriber Profile/LDAPv3](#)
  - [LDAP Server Schema](#)
  - [Configuration Workflow for Extended Web Services Subscriber Profile/LDAPv3](#)
  - [Management for Extended Web Services Subscriber Profile/LDAPv3](#)
  - [Provisioning for Extended Web Services Subscriber Profile/LDAPv3](#)
  - [Reference: Attributes and Operations for Extended Web Services Subscriber Profile/LDAPv3](#)

## Extended Web Services Subscriber Profile/LDAPv3

All subscriber profile related operations are handed off to network nodes that accept LDAP queries according to LDAPv3. The decision concerning which node in the LDAP directory should be used to perform the query is decided in runtime based on configuration settings. The data that is handed back to the application that initiated the Subscriber Profile query is filtered using the result filter mechanism in the service provider group and application group SLAs. See description of `<resultRestrictions>` in section [Defining Service Provider Group and Application Group SLAs](#) in *Managing Accounts and SLAs*

A connection pool is used for connections to the LDAP server. The connection pool is shared among all plug-in instances, and any configuration settings related to this pool or schema updates are broadcast to all plug-in instances in the cluster.

**Note:** To make any configuration change take effect, [Operation: updateLDAPSettings](#) must be used.

To see a	Refer to
Detailed list of necessary for managing and configuring the plug-in instance	<a href="#">Properties for Extended Web Services Subscriber Profile/LDAPv3</a>
Configuration workflow	<a href="#">Configuration Workflow for Extended Web Services Subscriber Profile/LDAPv3</a>
List of operations and attributes related to management and provisioning	<a href="#">Management for Extended Web Services Subscriber Profile/LDAPv3</a> <a href="#">Provisioning for Extended Web Services Subscriber Profile/LDAPv3</a>
Reference of management attributes and operations	<a href="#">Reference: Attributes and Operations for Extended Web Services Subscriber Profile/LDAPv3</a>

## Properties for Extended Web Services Subscriber Profile/LDAPv3

Property	Description
Managed object in Administration Console	<domain name>->WLNG-><server name>->Communication Services-><plug-in instance ID>
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName is same as plug-in instance ID. Type=com.bea.wlcp.wlng.plugin.subscriberprofile.ldap.managedplugin.management.SubscriberProfileMBean
Network protocol plug-in service ID	Plugin_ews_subscriber_profile_ldap
Network protocol plug-in instance ID	The ID is given when the plug-in instance is created, see <a href="#">Managing and Configuring the Plug-in Manager</a> .
Supported Address Scheme	tel id imsi ipv4
North interface	com.bea.wlcp.wlng.ews.plugin.SubscriberProfilePlugin
Service type	SubscriberProfile
Exposes to the service communication layer a Java representation of:	Extended Web Services Subscriber Profile

Property	Description
Interfaces with the network nodes using:	LDAP
Deployment artifacts	ews_subscriber_profile_service.jar and Plugin_ews_subscriber_profile_ldap.jar, packaged in wlng_nt_subscriber_profile_ews.ear ews_subscriber_profile.war, packaged in wlng_at_subscriber_profile_ews.ear

## LDAP Server Schema

A schema is used for constructing queries, see [Listing 24-1](#).

### Listing 24-1 LDAP Query schema XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="LdapConfig">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Keys" type="KeySet" minOccurs="1"
maxOccurs="unbounded"/>
        <xs:element name="LdapObject" type="LdapObject" minOccurs="1"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="KeyObject">
    <xs:sequence>
```

```

        <xs:element name="uriScheme" type="xs:string" minOccurs="1"
maxOccurs="1"/>

        <xs:element name="addressKeyName" type="xs:string" minOccurs="1"
maxOccurs="1"/>

        <xs:element name="objectKeyName" type="xs:string" minOccurs="0"
maxOccurs="1"/>

        <xs:element name="objectKeyValue" type="xs:string" minOccurs="0"
maxOccurs="1"/>

    </xs:sequence>

    <xs:attribute name="id" type="xs:string" use="optional"/>
</xs:complexType>

<xs:complexType name="KeySet">
    <xs:sequence>
        <xs:element name="Key" type="KeyObject" minOccurs="1"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="LdapObject">
    <xs:sequence>
        <xs:element name="ObjectKeySet" type="xs:string" minOccurs="0"
maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"/>
    <xs:attribute name="keyName" type="xs:string" use="required"/>
    <xs:attribute name="keyValue" type="xs:string" use="required"/>
</xs:complexType>
</xs:schema>

```

---

The document consists of the following elements:

- **LdapObject**, holder of a KeySet.
- **KeySet**, defines a collection of KeyObjects. Sets of keys are used since there may be several ways to reach a certain node in the tree. One LDAP plug-in instance can be configured with several KeySets, and can provide the link between the search key in the Extended Web Services Interface and the LDAP tree.
- **KeyObject**, defines an entry point to the LDAP tree, and provides the link between the search key in the Extended Web Services Interface and the LDAP tree.

**Table 24-1 LDAP server Schema**

Object	Element	Description
LdapObject	ObjectKeySet	Defines the KeySet through which it can be reached. Refers to the id attribute of a defined KeySet.
	id	The identity of the LdapObject. Can be referenced from other LdapObjects via the ParentObjectId field.
	keyName	The name of the key through which the LdapObject can be reached.
	keyValue	The value of the key through which the LdapObject can be reached.
KeyObject	uriScheme	Defines the URI scheme of the address for which this key applies.
	addressKeyName	Defines the key name with which the address value is associated.
	objectKeyName	This element provides the possibility to define the addressing key of a possible tree node above the node which is reached by the address key (i.e. like the domain object in the 3DS directory information tree).
	objectKeyValue	See objectKeyName, this element defines the value of the key.
	id	The identity of the key. Only used for descriptive purposes.



**Table 24-1 LDAP server Schema**

Object	Element	Description
KeySet	Key	All keys in the KeySet
	id	The identity of the KeySet. Used when associating an LdapObject with a KeySet.

[Listing 24-2](#) is an example of how a directory information tree is built up using the above schema.

**Listing 24-2 Example of LDAP server schema**

```
<?xml version="1.0" encoding="UTF-8"?>
<LdapConfig xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation='sp_config.xsd'>
  <Keys id="myKeys">
    <Key id="misidnKey">
      <uriScheme>tel</uriScheme>
      <addressKeyName>msisdn</addressKeyName>
      <objectKeyName>domainName</objectKeyName>
      <objectKeyValue>msisdnD</objectKeyValue>
    </Key>
    <Key id="imsiKey">
      <uriScheme>imsi</uriScheme>
      <addressKeyName>imsi</addressKeyName>
      <objectKeyName>domainName</objectKeyName>
      <objectKeyValue>imsiD</objectKeyValue>
    </Key>
    <Key id="subscriberIdKey">
      <uriScheme>id</uriScheme>
```

```
<addressKeyName>id</addressKeyName>
<objectKeyName>domainName</objectKeyName>
<objectKeyValue>subsD</objectKeyValue>
</Key>
<Key id="ipv4Key">
  <uriScheme>ipv4</uriScheme>
  <addressKeyName>ipv4Addr</addressKeyName>
  <objectKeyName>domainName</objectKeyName>
  <objectKeyValue>ipv4D</objectKeyValue>
</Key>
</Keys>
<LdapObject id="mySchema" keyName="serviceName" keyValue="mySchema">
  <ObjectKeySet>myKeys</ObjectKeySet>
</LdapObject>
</LdapConfig>
```

---

## Configuration Workflow for Extended Web Services Subscriber Profile/LDAPv3

Below is an outline for configuring an Extended Web Service Subscriber Profile/LDAPv3 network protocol plug-in instance:

1. Create one or more instances of the plug-in service: see [Managing and Configuring the Plug-in Manager](#). Use the plug-in service ID as detailed in [Properties for Extended Web Services Subscriber Profile/LDAPv3](#).
2. Using the Management Console or an MBean browser, select the MBean for the plug-in instance. The MBean display name is the same as the plug-in instance ID given when the plug-in instance was created.
3. Define the characteristics of the LDAP server to connect to by specifying:

[Attribute: Host](#)

[Attribute: Port](#)

[Attribute: BaseDN](#)

[Attribute: AuthDN](#)

[Attribute: AuthPassword](#)

4. The schema to use, using either:
  - [Attribute: Schema](#) or [Operation: updateSchemaURL](#)
  - See [LDAP Server Schema](#) for a description of the schema and [Configuration Workflow for Extended Web Services Subscriber Profile/LDAPv3](#) for a description of the mappings.
5. Define the connection pool characteristics for the connection:
  - [Attribute: MinConnections](#)
  - [Attribute: MaxConnections](#)
  - [Attribute: ConnTimeout](#)
  - [Attribute: RecoverTimerInterval](#)
6. Set up the routing rules to the plug-in instance: see [Configuring the Plug-in Manager](#). Use the plug-in instance ID and address schemes detailed in [Properties for Extended Web Services Subscriber Profile/LDAPv3](#).
7. If desired, create and load a node SLA, see:
  - [Defining Global Node and Service Provider Group Node SLAs](#)
  - [Managing Application SLAs](#)

Move on to the provisioning of service provider accounts and application accounts.

## Management for Extended Web Services Subscriber Profile/LDAPv3

No specific management operations, except for [Operation: updateLDAPSettings](#) which must be used in order to update the LDAP connection pool after changing any of the following attributes:

- [Attribute: MinConnections](#)
- [Attribute: MaxConnections](#)

- [Attribute: ConnTimeout](#)
- [Attribute: RecoverTimerInterval](#)

## Provisioning for Extended Web Services Subscriber Profile/LDAPv3

If the results from the LDAP query should be filtered, use the service provider group and application group SLAs. See description of `<resultRestrictions>` in section [Defining Service Provider Group and Application Group SLAs](#) in *Managing Accounts and SLAs*.

## Reference: Attributes and Operations for Extended Web Services Subscriber Profile/LDAPv3

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

- [Attribute: AuthDN](#)
- [Attribute: AuthPassword](#)
- [Attribute: BaseDN](#)
- [Attribute: ConnTimeout](#)
- [Attribute: Host](#)
- [Attribute: LDAPConnectionStatus](#)
- [Attribute: MaxConnections](#)
- [Attribute: MinConnections](#)
- [Attribute: RecoverTimerInterval](#)
- [Attribute: Port](#)
- [Attribute: Schema](#)
- [Operation: updateLDAPSettings](#)
- [Operation: updateSchemaURL](#)

## Attribute: Host

Scope: Cluster

Unit: n/a

Format: String

Specifies the host name or IP address of the LDAP server to connect to.

Examples:

`myldapserver.mycompany.org`

`192.168.0.14`

## Attribute: Port

Scope: Cluster

Unit: n/a

Format: int

Specifies the port number of the LDAP server to connect to.

## Attribute: BaseDN

Scope: Cluster

Unit: n/a

Format: String

Specifies the base DN (Distinguished name) for the LDAP database in use.

Example:

`o=acompany,c=uk`

## Attribute: AuthDN

Scope: Cluster

Unit: n/a

Format: String

Specifies the authentication user name (distinguished name) for the LDAP server.

Example:

```
cn=admin,o=acompany,c=uk
```

## Attribute: AuthPassword

Scope: Cluster

Unit: n/a

Format: String

Specifies the password associated with the [Attribute: AuthDN](#).

## Attribute: ConnTimeout

Scope: Cluster

Unit: Seconds

Specifies the maximum time to wait for an LDAP connection to be established. If the related timer expires, a retry is performed: see [Attribute: RecoverTimerInterval](#).

Any change to this setting must be followed by [Operation: updateLDAPSettings](#).

## Attribute: MinConnections

Scope: Cluster

Unit: n/a

Format: int

Specifies the minimum number of connections to establish using connections from the LDAP connection pool.

Any change to this setting must be followed by [Operation: updateLDAPSettings](#).

## Attribute: MaxConnections

Scope: Cluster

Unit: n/a

Format: int

Specifies the maximum number of connections in the LDAP connection pool.

Any change to this setting must be followed by [Operation: updateLDAPSettings](#).

## Attribute: Schema

Scope: Cluster

Unit: n/a

Format: String

The LDAP schema to use.

## Attribute: LDAPConnectionStatus

Read-only.

Scope: Cluster

Unit: n/a

Format: String enumeration:

**Table 24-2 Status of the connection to the LDAP server**

Status	Description
active	The connection is active. The plug-in instance accepts requests.
update_pending	The connection is temporarily unavailable due to an update of the configuration settings. The plug-in instance does not accept requests.
deactive	The connection is inactive. The plug-in instance does not accept requests. Reasons for this entering this state includes: <ul style="list-style-type: none"> <li>• Missing or incorrect configuration.</li> <li>• LDAP server is unreachable.</li> <li>• Internal errors.</li> </ul>

## Attribute: RecoverTimerInterval

Scope: Cluster

Unit: Seconds

Specifies the time to wait before performing an LDAP connection retry after an LDAP connection error. Should be at least twice the time defined in [Attribute: ConnTimeout](#).

Any change to this setting must be followed by [Operation: updateLDAPSettings](#).

## Operation: updateLDAPSettings

Scope: Cluster

Refreshes the LDAP connection pool to use the new configuration.

During the update, the LDAP connection is temporary unavailable and the connection status is `update_pending`. See [Status of the connection to the LDAP server](#).

Signature:

```
updateLDAPSettings()
```

**Table 24-3** updateLDAPSettings

updateLDAPSettings	
Parameter	Description
-	-

## Operation: updateSchemaURL

Scope: Cluster

Updates the schema to use when doing lookups in the LDAP database.

During the update, the LDAP connection is temporary unavailable and the connection status is `update_pending`. See [Status of the connection to the LDAP server](#).

Signature:

```
updateSchemaURL(SchemaURL : String)
```



**Table 24-4 updateSchemaURL**

<b>updateSchemaURL</b>	
<b>Parameter</b>	<b>Description</b>
SchemaURL	URL to the LDAP database schema. Examples: file:///d:/ldap/schema.xml (Windows systems) file://ldap/schema.xml (UNIX systems)

## Managing and Configuring Subscriber Profile Communication Services

# Managing and Configuring WAP Push Communication Services

The following section describes configuration and maintenance attributes and operations for the Extended Web Services WAP Push communication service. It also provides a workflow for the configuration:

- [Extended Web Services WAP Push/PAP](#)
  - [Configuration Workflow for Extended Web Services WAP Push/PAP](#)
  - [Management for Extended Web Services WAP Push/PAP](#)
  - [Provisioning for Extended Web Services WAP Push/PAP](#)
  - [Reference: Attributes and Operations for WAP Push/PAP](#)

## Extended Web Services WAP Push/PAP

All WAP Push related operations are handed off to network nodes that accept and forward WAP Push messages to end user terminals via a Push Proxy Gateway (PPG).

To see a	Refer to
Detailed list of necessary for managing and configuring the plug-in instance	<a href="#">Properties for Extended Web Services WAP Push/PAP</a>
Configuration workflow	<a href="#">Configuration Workflow for Extended Web Services WAP Push/PAP</a>

To see a	Refer to
List of operations and attributes related to management and provisioning	<a href="#">Management for Extended Web Services WAP Push/PAP</a> <a href="#">Provisioning for Extended Web Services WAP Push/PAP</a>
Reference of management attributes and operations	<a href="#">Reference: Attributes and Operations for WAP Push/PAP</a>

## Properties for Extended Web Services WAP Push/PAP

Property	Description
Managed object in Administration Console	<domain name>->WLNG-><server name>->Communication Services-><plug-in instance ID>
MBean	Domain=com.bea.wlcp.wlmg Name=wlmg_nt InstanceName is same as plug-in instance ID. Type=com.bea.wlcp.wlmg.plugin.pushmessage.pap.management.PushMessagePAPMBean
Network protocol plug-in service ID	Plugin_ews_push_message_pap
Network protocol plug-in instance ID	The ID is given when the plug-in instance is created, see <a href="#">Managing and Configuring the Plug-in Manager</a> .
Supported Address Scheme	tel wapuser
North interface	com.bea.wlcp.wlmg.ews.plugin.PushMessagePlugin com.bea.wlcp.wlmg.ews.callback.PushMessageNotificationCallback
Service type	PushMessage

Property	Description
Exposes to the service communication layer a Java representation of:	Extended Web Services WAP Push
Interfaces with the network nodes using:	Push Access Protocol (PAP), 2.0. WAP-247-PAP-20010429-a
Deployment artifacts	ews_push_message_service.jar, Plugin_ews_push_message_pap.jar, and ews_push_message_pap.war, packaged in wlng_nt_push_message_ews.ear  ews_push_message.war and ews_push_message_callback.jar, packaged in wlng_at_push_message_ews.ear

## Configuration Workflow for Extended Web Services WAP Push/PAP

Below is an outline for configuring an Extended Web Service WAP Push/PAP network protocol plug-in instance:

1. Create one or more instances of the plug-in service: see [Managing and Configuring the Plug-in Manager](#). Use the plug-in service ID as detailed in [Properties for Extended Web Services WAP Push/PAP](#).
2. Using the Management Console or an MBean browser, select the MBean for the plug-in instance. The MBean display name is the same as the plug-in instance ID given when the plug-in instance was created.
3. Define the characteristics of the PPG server to connect to by specifying:
  - [Attribute: PPGNotificationURL](#)
  - [Attribute: PPGURL](#)
4. Specify heartbeat behavior, see [Configuring Heartbeats](#)
5. Set up the routing rules to the plug-in instance: see [Configuring the Plug-in Manager](#). Use the plug-in instance ID and address schemes detailed in [Properties for Extended Web Services WAP Push/PAP](#).
6. If desired, create and load a node SLA, see:

[Defining Global Node and Service Provider Group Node SLAs](#)

[Managing Application SLAs](#)

Move on to the provisioning of service provider accounts and application accounts.

## Management for Extended Web Services WAP Push/PAP

No specific management operations.

## Provisioning for Extended Web Services WAP Push/PAP

No specific provisioning operations.

## Reference: Attributes and Operations for WAP Push/PAP

Below is a list of attributes for configuration and maintenance:

- [Attribute: PPGNotificationURL](#)
- [Attribute: PPGURL](#)

### Attribute: PPGNotificationURL

Scope: Server

Unit: n/a

Format: String

Specifies the URL of the plug-in instance. Used by the Push Proxy Gateway (PPG) to send notifications of results to the plug-in instance.

### Attribute: PPGURL

Scope: Cluster

Unit: n/a

Format: String

Specifies the URL of the Push Proxy Gateway (PPG) the plug-in instance uses.

# Managing and Configuring SOAP2SOAP Communication Services

The following section describes configuration and maintenance attributes and operations for SOAP2SOAP communication services. The plug-ins for SOAP2SOAP communication services are created using the Platform Development Studio. The sections also provide a workflow for the configuration:

- [SOAP2SOAP Communication Services](#)
  - [About SOAP2SOAP Communication Services](#)
  - [Configuration Workflow for SOAP2SOAP Plug-ins](#)
  - [Provisioning Workflow for SOAP2SOAP Communication Services](#)
  - [Reference: Attributes and Operations for SOAP2SOAP Plug-ins](#)

## SOAP2SOAP Communication Services

This section contains a description of the configuration attributes and operations available for SOAP2SOAP-type of plug-in instances.

To see a	Refer to
Detailed list of necessary for managing and configuring a plug-in instance	<a href="#">Properties for SOAP2SOAP Plug-ins</a>
Configuration workflow	<a href="#">Configuration Workflow for SOAP2SOAP Plug-ins</a>
List of operations and attributes related to management and provisioning	<a href="#">Provisioning Workflow for SOAP2SOAP Communication Services</a>
Reference of management attributes and operations	<a href="#">Reference: Attributes and Operations for SOAP2SOAP Plug-ins</a>

## About SOAP2SOAP Communication Services

SOAP2SOAP communication services are developed using the Platform Development Studio.

Based on a set of service WSDLs and callback WSDLs, a SOAP2SOAP communication service acts as a proxy service, and provides the functionality provided by the Network Gatekeeper container services, such as:

- SLA enforcement
- EDR, CDR, and Alarms
- Authentication, access control, and accounting

SOAP2SOAP plug-in services can be instantiated using the plug-in manager. The plug-in instances process traffic and connects to individual network nodes. The instances are managed independently of each other.

For application-initiated requests, all requests are routed to the network node defined for the plug-in instance.

For network-triggered requests, the network-node should distinguish which application instance the request is targeted to by adding the application instance ID to the URL:

```
http://<WLNNG IP Address>:<port>/<context-root>_nt/<plug-in instance ID>_nt/<application instance ID>
```



The endpoint at which the application instance has implemented the Web Service is provisioned: see [Provisioning Workflow for SOAP2SOAP Communication Services](#).

## Properties for SOAP2SOAP Plug-ins

Property	Description
Managed object in Administration Console	<domain name>->WLNG-><server name>->Communication Services-><plug-in instance ID>
MBean	Domain=com.bea.wlcp.wlmg Name=wlngt InstanceName is same as plug-in instance ID Type=com.bea.wlcp.wlmg.httpproxy.management.HTTPProxyMBean
Network protocol plug-in service ID	Defined when generating the SOAP2SOAP plug-in using the Platform Development Studio.
Network protocol plug-in instance ID	The ID is given when the plug-in instance is created, see <a href="#">Managing and Configuring the Plug-in Manager</a> .
Supported Address Scheme	Given when the plug-in was generated using the Platform Development Studio.
North interfaces	Derived from the package name of the network protocol plug-in, as given when the plug-in was generated, and the name of the interface as defined in the WSDL. For application-initiated request: <package name>.plugin.<Interface name>Plugin For network-triggered requests: <package name>.callback.<Interface name>Callback See see <a href="#">Managing and Configuring the Plug-in Manager</a> for information on how to list the interfaces.
Service type	Given when the plug-in was generated using the Platform Development Studio.
Exposes to the service communication layer a Java representation of:	Depends on the WSDLs used.

Property	Description
Interfaces with the network nodes using:	The same protocol as exposed by the application-facing interfaces.
Deployment artifacts	<p>&lt;communication service identifier&gt;_&lt;protocol&gt;_plugin.jar, &lt;communication service identifier&gt;_service.jar and &lt;communication service identifier&gt;_callback.war, packaged in wlng_nt_&lt;communication service&gt;.ear</p> <p>&lt;communication service identifier&gt;_callback.jar and &lt;communication service identifier&gt;.war, packaged in wlng_at_&lt;communication service identifier&gt;.ear</p> <p>&lt;communication service identifier&gt; and &lt;protocol&gt; was given in the Eclipse wizard when the communication service was generated.&gt;</p>

## Configuration Workflow for SOAP2SOAP Plug-ins

Below is an outline for configuring the plug-in using the Network Gatekeeper Administration Console:

1. Create one or more instances of the plug-in service: see [Managing and Configuring the Plug-in Manager](#). Use the plug-in service ID as detailed in [Properties for SOAP2SOAP Plug-ins](#).
2. Using the Management Console or an MBean browser, select the MBean for the plug-in instance. The MBean display name is the same as the plug-in instance ID given when the plug-in instance was created.
3. Configure the authentication credentials to be used by the plug-in towards the network node:
  - Attribute: [UserName](#)
  - Attribute: [Password](#)
4. Specify which authentication method to use towards the network node:
  - Attribute: [HttpBasicAuthentication](#)
  - Attribute: [UserNameTokenAuthentication](#)
5. Specify the URL of the network node to connect to:
  - Attribute: [UserNameTokenAuthentication](#)

6. Specify heartbeat behavior, see [Configuring Heartbeats](#).
7. Set up the routing rules to the plug-in instance: see [Configuring the Plug-in Manager](#). Use the plug-in instance ID and address schemes detailed in [Properties for SOAP2SOAP Plug-ins](#).

**Note:** If the SOAP2SOAP plug-in is the only plug-in for the service enabler, routing based on destination address is not applicable. All requests will be routed to the plug-in instance. If there are other, non-SOAP2SOAP, plug-ins in the service enabler, the routing applies.

8. Provide the administrator of the network node server with the URL to which it should deliver network-triggered requests. The default URL is

```
http://<IP Address>:<port>/<context-root>_nt/<plug-in instance ID>/<application instance ID>
```

default `<context-root>` is the communication service ID, as specified when the communication service was generated.

`<plug-in instance ID>` is the ID of the plug-in instance. The network node is assumed to specify which application instance the request is targeted to by adding the application instance ID as the suffix in the URL.

9. If desired, create and load a node SLA, see:

[Defining Global Node and Service Provider Group Node SLAs](#)

[Managing Application SLAs](#)

Continue with the provisioning of service provider accounts and application accounts.

## Provisioning Workflow for SOAP2SOAP Communication Services

For each application that uses SOAP2SOAP communication services and supports network-triggered requests, a mapping must be set up between the application instance ID and the URL for the Web service this application instance implements. Use the following operations to manage the callback URLs for the application instances:

- [Operation: addApplicationEndPoint](#)
- [Operation: getApplicationEndPoint](#)
- [Operation: listApplicationEndPoints](#)
- [Operation: removeApplicationEndPoint](#)

## Reference: Attributes and Operations for SOAP2SOAP Plug-ins

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

- [Attribute: HttpBasicAuthentication](#)
- [Attribute: NetworkEndpoint](#)
- [Attribute: Password](#)
- [Attribute: UserName](#)
- [Attribute: UserNameTokenAuthentication](#)
- [Operation: addApplicationEndPoint](#)
- [Operation: getApplicationEndPoint](#)
- [Operation: listApplicationEndPoints](#)
- [Operation: removeApplicationEndPoint](#)

### Attribute: HttpBasicAuthentication

Scope: Cluster

Unit: n/a

Format: Boolean

Specifies whether HTTP Basic Authentication shall be used when authenticating with the network node. Enter:

- `true` to use HTTP Basic Authentication
- `false` to not.

### Attribute: NetworkEndpoint

Scope: Cluster

Unit: n/a

Format: String

Specifies the URL to the network node.

## Attribute: Password

Scope: Cluster

Unit: n/a

Format: String

Specifies the password to use when connecting to the network node.

## Attribute: UserName

Scope: Cluster

Unit: n/a

Format: String

Specifies the username to use when connecting to the network node.

## Attribute: UserNameTokenAuthentication

Scope: Cluster

Unit: n/a

Format: Boolean

Specifies whether Web Services Security Username Token shall be used when authenticating with the network node.

- `True` to use UserName Token Authentication
- `False` to not.

## Operation: addApplicationEndPoint

Scope: Cluster

Adds the URL to which network-triggered requests per application instance should be forwarded. The application is assumed to implement the callback web service on this URL.

Signature:

```
addApplicationEndPoint(AppinstanceId: String, CallbackUrl: String)
```

**Table 26-1 addApplicationEndPoint**

<b>addApplicationEndPoint</b>	
<b>Parameter</b>	<b>Description</b>
AppinstanceId	The application instance ID associated with the callback URL.
CallbackUrl	The URL to the implementation of the application instance's implementation of the callback.

## Operation: **getApplicationEndPoint**

Scope: Cluster

Displays the URL to which network-triggered requests for a given application instance should be forwarded.

Signature:

```
getApplicationEndPoint(AppinstanceId: String)
```

**Table 26-2 getApplicationEndPoint**

<b>getApplicationEndPoint</b>	
<b>Parameter</b>	<b>Description</b>
AppinstanceId	The application instance ID associated with the callback URL.

## Operation: **listApplicationEndpoints**

Scope: Cluster

Displays a list of all registered callback URLs.

Signature:

```
listApplicationEndpoints()
```

**Table 26-3 listApplicationEndpoints**

<b>listApplicationEndpoints</b>	
<b>Parameter</b>	<b>Description</b>
-	-

## Operation: removeApplicationEndPoint

Scope: Cluster

Removes the URL to which the network-triggered requests for a given application instance are forwarded.

Signature:

```
removeApplicationEndPoint(AppinstanceId: String)
```

**Table 26-4 removeApplicationEndPoint**

<b>removeApplicationEndPoint</b>	
<b>Parameter</b>	<b>Description</b>
AppinstanceId	The application instance ID associated with the callback URL.

## Managing and Configuring SOAP2SOAP Communication Services



# Configuring Heartbeats

The following section describes how to configure heartbeats for HTTP-based stateless network protocol plug-ins.

- [Configuration and Management](#)
- [Reference: Attributes and Operations for HeartbeatConfiguration](#)

## Introduction

The heartbeat functionality performs heartbeat checks towards HTTP based network nodes on behalf of a plug-in. When a heartbeat fails, the plug-in is set to state INACTIVE. The heartbeat functionality will continue trying to connect to the node and when a positive answer is received the plug-in re-enters state ACTIVE.

The following network protocol plug-ins use this functionality:

- Parlay X 2.1 Short Messaging/MLP
- Extended Web Services WAP Push/PAP
- Parlay X 2.1 Multimedia Messaging/MM7

## Configuration and Management

The heartbeat OAM functionality is shared among the plug-ins. They all use the same MBean, `com.bea.wlcp.wlmg.heartbeat.management.HeartbeatMBean`. But the result is rendered

per plug-in (one instance is displayed per plug-in). It appears slightly differently depending how to the MBean is accessed:

- Using the Management Console, one instance of the OAM is displayed in connection with the plug-in it is used by, so the management attributes for the heartbeat module are reached from the service:
  - Plugin\_px21\_multimedia\_messaging->HeartbeatConfiguration
  - Plugin\_ews\_push\_message\_pap->HeartbeatConfiguration
  - Plugin\_px21\_terminal\_location\_mlp->HeartbeatConfiguration
- In an MBean browser, such as JConsole, one instance of the MBean is displayed using the the same ObjectName (at the same hierarchical level) as the plug-in it is used by.

**Note:** You need to configure heartbeat attributes for *all* of the above mentioned plug-ins that are going to use this functionality. Only the heartbeat attributes for the related plug-in are displayed in the OAM.

1. Specify if heartbeats should be enabled for the associated plug-in or not in [Attribute: Enabled](#).
2. If heartbeats are enabled, define:

[Attribute: ContentMatch](#)

[Attribute: Interval](#)

[Attribute: NetworkServiceUrl](#)

## Reference: Attributes and Operations for HeartbeatConfiguration

Managed object: HeartbeatConfiguration

MBean: om.bea.wlcp.wlmg.heartbeat.management.HeartbeatMBean

Below is a list of attributes for configuration:

- [Attribute: ContentMatch](#)
- [Attribute: Enabled](#)
- [Attribute: Interval](#)
- [Attribute: NetworkServiceUrl](#)

- [Attribute: Status \(r\)](#)

## Attribute: ContentMatch

Scope: Cluster

Unit: n/a

Format: String

The content for matching the returned heartbeats. If the returned string differs from this, the heartbeat is assumed to have failed. Leave this empty to match any content.

## Attribute: Enabled

Scope: Cluster

Unit: n/a

Format: Boolean

Specifies if the heartbeat functionality is applied for the plug-in.

## Attribute: Interval

Scope: Cluster

Unit: seconds

Format: integer

Specifies the time interval between heartbeats.

## Attribute: NetworkServiceUrl

Scope: Cluster

Unit: n/a

Format: String

Specifies the URL to which heartbeats are sent.

## Attribute: Status (r)

Scope: Cluster

Unit: n/a

## Configuring Heartbeats

Format: boolean

Read-only.

If true, there was a successful response to the last heartbeat. If the previous Status was false, the plug-in state is altered from INACTIVE to ACTIVE.

If false, there was no or an incorrect response to the last heartbeat. In this case the associated plug-in is set state INACTIVE and an alarm is generated.

# Managing and configuring Shortcode mappings

The following section describes how to configure shortcode mappings for incoming messages to Network Gatekeeper.

- [Configuration and Management](#)
- [Reference: Attributes and Operations for ShortCodeMapper](#)

## Introduction

The Shortcode Mapper is shared between the following plug-ins:

- Parlay X 2.1 Short Messaging/SMPP
- Parlay X 2.1 Multimedia Messaging/MM7

The Shortcode mapper maps network-triggered messages (mobile originated) with a given destination address, the *original destination address*, to another destination address, the *translated destination address*.

The original destination address can be expressed as a pattern, that is, a regular expression, which means that a range, or set, of original destination addresses can be translated to one single translated destination address.

This is useful when having applications that are triggered by a range of phone numbers, for example, a range of phone numbers like 2345600 through 2345699. Using the functionality available in the Parlay X 2.1 Short Messaging interface the application would have to call `startSmsNotification` 100 times, while setting up a short code translation mapping where the original destination addresses are expressed as 23456?? and the translated destination address is

set to 1234 means that only a single call to `startSmsNotification` using the address 1234 is required.

## Configuration and Management

All configuration and management is performed in the managed object `ShortCodeMapper`.

The `ShortCodeMapper` OAM functionality is shared between the plug-ins: they reuse the same MBean. Rendering however is per plug-in (one instance is displayed per plug-in). It appears slightly differently depending how to the MBean is accessed:

- Using the Administration Console, one instance of the MBean is displayed in connection with the plug-in it is used by, so the management attributes for the `ShortCodeMapper` module are reached from the service:
  - `Plugin_px21_multimedia_messaging_mm7->ShortCodeMapper`
  - `Plugin_px21_short_messaging_smpp->ShortCodeMapper`
- In an MBean browser, such as JConsole, one instance of the MBean is displayed using the the same `ObjectName` (at the same hierarchical level) as the plug-in it is used by.

**Note:** You need to configure `ShortCodeMapper` attributes for *all* of the above mentioned plug-ins that are going to use this functionary. Only the attributes for the related plug-in are displayed in the OAM.

## Management operations

Management of shortcodes includes:

Operation: [addShortCodeMapping](#)

Operation: [listShortCodeMappings](#)

Operation: [removeShortCodeMapping](#)

## Reference: Attributes and Operations for ShortCodeMapper

Managed object: `ShortCodeMapper`

MBean: `com.bea.wlcp.wlng.shortcode.management.ShortCodeMapperMBean`

Below is a list of operations for configuration:

- [Operation: addShortCodeMapping](#)
- [Operation: listShortCodeMappings](#)
- [Operation: removeShortCodeMapping](#)

## Operation: addShortCodeMapping

Scope: Cluster

Translates the destination address of a network-triggered message to a translated destination address. The translated destination address is used when matching the message with notification registrations. This makes it possible for an application to register for a notification from a single shortcode and receive messages sent to a range of short codes (original destination addresses).

**Note:** Short code translation takes place before mapping a mobile originated message to registered application notifications. If two or more short code translations match an original destination number the best match, the most specific, is chosen.

An identifier for the short code translation is returned.

Signature:

```
addShortCodeMapping(orig_dest_pattern: String, trans_dest_addr: String)
```

**Table 28-1** addShortCodeMapping

addShortCodeMapping	
Parameter	Description
orig_dest_pattern	Pattern that is matched against the original destination address as of a message received from the network. Regular expression. Addresses must be specified with the prefix “tel:”, for example: ^tel:6.*
trans_dest_addr	The resulting translated address. Addresses must be specified with the prefix “tel:”.

## Operation: listShortCodeMappings

Scope: Cluster

## Managing and configuring Shortcode mappings

Displays a list of shortcode mappings.

Signature:

```
listShortCodeMappings()
```

**Table 28-2 listShortCodeMappings**

listShortCodeMappings	
Parameter	Description
-	-

## Operation: removeShortCodeMapping

Scope: Cluster

Removes a previously added shortcode mapping. Both fields must match.

Signature:

```
removeShortCodeMapping(OriginalDestination: String, MappedDestination:  
String)
```

**Table 28-3 removeShortCodeMapping**

removeShortCodeMapping	
Parameter	Description
OriginalDestination	Registered original destination pattern.
MappedDestination	Registered translated destination address.



# Configuring WebLogic SIP Server Integration

The following section describes how to configure the WebLogic Network Gatekeeper and WebLogic SIP Server integration.

- [Introduction](#)
- [Configuration Workflow](#)
- [Reference: Attributes and operations -General](#)
- [Reference: Attributes and operations -Third Party Call Control](#)
- [Reference: Attributes and operations -Call Notification](#)
- [Reference: Attributes and operations -Presence](#)

## Introduction

When using the WebLogic SIP server and WebLogic Network Gatekeeper together, the integration points between them must be configured in order for requests to be correlated between the SIP Server parts and Network Gatekeeper network tier. The WebLogic SIP Server part of the configuration is described in this section. The Network Gatekeeper configuration is described in the description of the individual communication service.

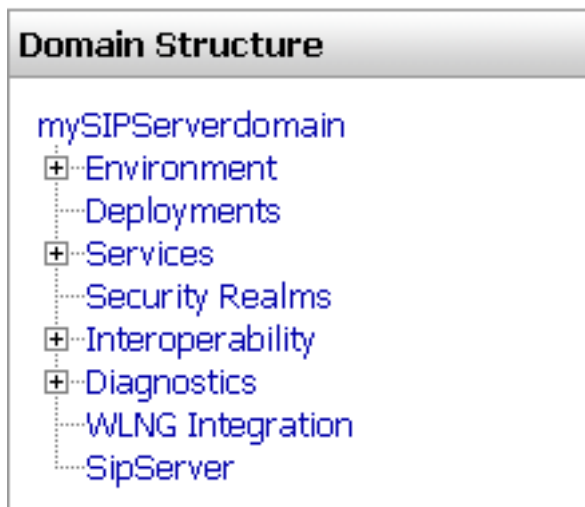
This is necessary when using the following network protocol plug-ins, since SIP Application sessions are used in the following communication services:

- Parlay X 2.1 ThirdParty Call

- Parlay X 2.1 Call Notification
- Parlay X 2.1 Presence

See [BEA WebLogic SIP Server](#) documentation for information on how to configure and manage WebLogic SIP Server.

There is a Network Gatekeeper Integration management console in the SIP server, found under the node **WLNG Integration** under the **Domain Structure** in the SIP Server administration console.



**Note:** The necessary integration software must be installed in the SIP server, see section [Completing Post-Installation](#) in the *Network Gatekeeper Installation Guide*.

## Configuration Workflow

Configure the WebLogic SIP Server part of the Network Gatekeeper and WebLogic SIP Server integration:

1. In the Administration Console, for the 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 other attributes should not be changed.

## Reference: Attributes and operations -General

Below is a list of attributes and operations for configuration and maintenance of the general part of the Network Gatekeeper and SIP Server integration:

- [Attribute: WLNG JNDI URL](#)
- [Attribute: Username](#)
- [Attribute: Password](#)

### Attribute: WLNG JNDI URL

The URL to the JNDI provider in WebLogic Network Gatekeeper as configured during the domain configuration.

Example: t3://<host>:<port>

### Attribute: Username

A WebLogic Network Gatekeeper administrative user.

## Attribute: Password

The password associated with the WebLogic Network Gatekeeper administrative user.

## Reference: Attributes and operations -Third Party Call Control

Below is a list of attributes and operations for configuration and maintenance of the SIP Server part of the Parlay X 2.1 Third Party Call/SIP plug-in:

- [Attribute: Call Manager Class Name](#)
- [Attribute: SIP Servlet Name](#)
- [Attribute: Controller URI](#)

## Attribute: Call Manager Class Name

Class for that handles call management for Third Party call. Normally this field should not be edited.

## Attribute: SIP Servlet Name

The SIP servlet for Third Party Call. Normally this field should not be edited.

## 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.

## Reference: Attributes and operations -Call Notification

Below is a list of attributes and operations for configuration and maintenance of the SIP Server part of the Parlay X 2.1 Call Notification/SIP plug-in:

- [Attribute: Call Manager Class Name](#)
- [Attribute: SIP Servlet Name](#)

## Attribute: Call Notification Plugin JNDI Name

JNDI name for the Call Notification part of the Parlay X 2.1 Call Notification/SIP plug-in. Normally this field should not be edited.

## Attribute: Call Direction Plugin JNDI Name

JNDI name for the Call Direction part of the Parlay X 2.1 Call Notification/SIP plug-in. Normally this field should not be edited.

# Reference: Attributes and operations -Presence

Below is a list of attributes and operations for configuration and maintenance of the SIP Server part of the Parlay X 2.1 Presence/SIP plug-in:

- [Attribute: Suggested subscription lifetime](#)
- [Attribute: Presence Server address](#)

## 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>

## Configuring WebLogic SIP Server Integration

# Managing OSA/Parlay Gateway Connections using Parlay\_Access

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 Gateway Administrator](#)
- [Overall workflow when connecting to an OSA Gateway](#)
  - [Adding an OSA/Parlay 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 Parlay\\_Access](#)

## Understanding OSA/Parlay Gateway and account mappings

### Connection model

Network Gatekeeper communication services use an internal service, the Parlay\_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, 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/Parlay 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/Parlay 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/gwID`.
- An **OSA/Parlay client** represents the account in the OSA/Parlay Gateway. An OSA client has the following attributes:
  - OSA client application ID, made up 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/Parlay client mapping** maps an OSA/Parlay client with OSA/Parlay SCSes. There must be (at least) one OSA/Parlay client mapping per OSA SCS being used. If the communication service uses  $n$  OSA/Parlay SCSs,  $n$  Client Mappings must be defined. Three different models are possible for the OSA/Parlay Client Mapping:
  - a. The 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 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 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 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 Client. This means that every transaction originating from a specific



application results in a transaction in the OSA/Parlay 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/Parlay client mapping is a part of the provisioning chain in when setting up service provider and application accounts if the client mapping is of type b. or type c.

Each OSA/Parlay 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/Parlay Gateway is active and functional.
- **Inactive**, which means that there is no active connection. This may be because the 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 Gateway 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 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/Parlay 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 SCSes to use in the OSA/Parlay Gateway is done either as a part of the configuration flow for the communication services, 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 communication service.

## Adding an OSA/Parlay Gateway

An OSA/Parlay Gateway is the entity representing an OSA/Parlay Gateway. One or more OSA Gateway Connections can be associated with the OSA Gateway.

1. If authenticating using certificates, 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\_Parlay\_Access\_<communication service>**, 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. These 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.
  - c. The IOR is provided as a String.
2. Starting in the configuration and operations page for **Plugin\_Parlay\_Access\_<communication service>**, select:

If the IOR is provided as a file: use [Operation: addConnection](#)

If the IOR is provided as a String: use [Operation: addConnectionIOR](#)

3. Click **Invoke**.

The OSA Gateway Connection is created. An ID for the OSA Gateway Connection is returned.

## Creating an OSA client

The OSA client is the entity being used when creating the OSA client mapping.

1. If authenticating using certificates, 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 **Plugin\_Parlay\_Access\_<communication service>**, 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\_Parlay\_Access\_<communication service>**, 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 Parlay \_Access

Managed object: Container Services->Parlay\_Access\_<Communication Service>

Where Communication Service is one of:

- audio\_call\_px30
- call\_notification\_px30
- third\_party\_call\_px30

**Note:** There are three MBeans, one for each communication service of Parlay type. It does not matter which one you use, they all operate on the same data.

MBean: com.bea.wlcp.wlmg.parlay.access.ParlayAccessMBean

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

- [Attribute: EricssonAuthentication](#)
- [Operation: activateMapping](#)
- [Operation: addClient](#)
- [Operation: addConnection](#)
- [Operation: addConnectionIOR](#)
- [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

Scope: Cluster

Unit: n/a

Format: Boolean

Set to:

- True if connecting to an Ericsson OSA/Parlay Gateway
- False if connecting to other gateways.

## Operation: activateMapping

Scope: Cluster

Activates an existing mapping.

Signature:

```
activateMapping(id: String)
```

**Table 30-1 activateMapping**

activateMapping	
Parameter	Description
id	ID of the OSA/Parlay client mapping to activate. See <a href="#">Operation: listMappings</a> .

## Operation: addClient

Scope: Cluster

Adds an OSA/Parlay Client.

Signature:

```
addClient(osaClientAppId: String, clientKeyFile: String, clientCertFile:
String, clientKeyPwd: String, keystorePwd: String)
```

**Table 30-2 addClient**

<b>addClient</b>	
<b>Parameter</b>	<b>Description</b>
osaClientAppId	<p>The Enterprise Operator ID and Application ID registered for the OSA/Parlay Client in the OSA/Parlay Gateway. This value must be unique. The format is:</p> <p>&lt;Enterprise Operator&gt;\&lt;Application ID&gt;</p> <p>Example:</p> <p>myEntopId\myAppId</p>
clientKeyFile	<p>The directory path (including file name) to the private key for the OSA Client.</p> <p><b>Note:</b> This path is on the file system of the Network Gatekeeper network Tier server.</p> <p>Leave empty if not authenticating using certificates.</p>
clientCertFile	<p>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.</p> <p><b>Note:</b> This path is on the file system of the Network Gatekeeper Network Tier server.</p> <p>Leave empty if not authenticating using certificates.</p>
clientKeyPwd	<p>The password for the private key.</p> <p>Leave empty if not authenticating using certificates.</p>
keystorePwd	<p>The keystore's password as defined when configuring the WebLogic Network Gatekeeper, see <a href="#">Operation: setKeyStorePassword</a>.</p>

## Operation: addConnection

Scope: Cluster

Adds a connection to a Framework in the OSA/Parlay Gateway using a file that contains the name service IOR.

Signature:

```
addConnection(gwId: int, nsRef: String, nsName: String, initialRef: String,  
priority: int)
```

**Table 30-3 addConnection**

<b>addConnection</b>	
<b>Parameter</b>	<b>Description</b>
gwId	The ID of the OSA/Parlay Gateway, as returned when the OSA Gateway was created. See <a href="#">Operation: addGw</a> . Also see <a href="#">Operation: listGw</a> .
nsRef	The directory path (including file name) for the file containing the name service IOR.  Leave blank if initialRef is specified.
nsName	The name of the initial object in the name service. Example: <code>parlay_initial</code> .  Use path syntax to specify recursive naming contexts. Example: <code>/parlay/fw/parlay_initial</code>  Leave blank if initialRef is specified.
initialRef	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.
priority	Priority of this connection. Should be unique across all connections. The lower the number, the higher the priority.

## Operation: addConnectionIOR

Scope: Cluster

Adds a connection to a Framework in the OSA/Parlay Gateway using an IOR string.

Signature:



```
addConnectionIOR(gwId: int, ior: String, ns: String, priority: int)
```

**Table 30-4 addConnectionIOR**

<b>addConnection</b>	
<b>Parameter</b>	<b>Description</b>
gwId	The ID of the OSA/Parlay Gateway, as returned when the OSA Gateway was created. See <a href="#">Operation: addGw</a> . Also see <a href="#">Operation: listGw</a> .
ior	IOR string of either the NS or the initial object.
ns	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 IOR to the initial object is specified.
priority	Priority of this connection. Should be unique across all connections. The lower the number, the higher the priority.

## Operation: addGw

Scope: Cluster

Adds an OSA/Parlay Gateway to be used by the OSA/Parlay type plug-ins. More than one Gateway can be added.

Signature:

```
addGw(name: String, osaFwCert: String, reAuthWaitTime: int, keystorePwd: String)
```

Returns the ID for the OSA Gateway. This ID is used when creating an OSA/Parlay Gateway Connection, see [Operation: addConnection](#) and when creating an OSA/Parlay Client Mapping, see [Operation: addMapping](#).

**Table 30-5 addGw**

<b>addGw</b>	
<b>Parameters</b>	<b>Description</b>
gateway.name	Descriptive name of the OSA Gateway.
osaFwCert	The certificate to use when connecting to the OSA Gateway's Framework. The certificate is supplied by the OSA Gateway administrator. Leave empty if not authenticating using certificates.
reAuthWaitTime	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
keystorePwd	The password for the Network Gatekeeper keystore.

## Operation: addMapping

Scope: Cluster

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 30-6 AddMapping**

<b>addMapping</b>	
<b>Parameter</b>	<b>Description</b>
serviceProviderID	<p>ID of the service provider account the application is associated with.</p> <p><b>Note:</b> If left empty, the mapping will <i>not</i> be applied on service provider account and application account level.</p>
applicationID	<p>ID of the application account.</p> <p><b>Note:</b> If left empty, the mapping will <i>not</i> be applied on application account level.</p>
serviceType	<p>OSA/Parlay service type name (TpServiceTypeName) of the OSA/Parlay SCS to which the OSA Client is to be mapped.</p> <p>See the specification for the OSA/Parlay Framework for a list of recommended service type names.</p>
osaClientAppId	<p>The OSA/Parlay account's <code>clientAppID</code>, a string consisting of the <code>entOpId</code> followed by <code>\</code>, followed by the <code>appId</code>. Example: <code>sp1\app1</code>.</p> <p>The <code>entOpId</code> and <code>appId</code> is provided by the OSA Gateway administrator.</p>
properties	<p>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.</p> <p>The properties are specified as a space separated list in the following way:  <code>&lt;propname1&gt; &lt;propval1&gt; &lt;propname2&gt; &lt;propval2&gt;</code></p> <p>The properties varies between OSA/Parlay Gateway implementations.</p>
authType	<p>Authentication type to be used. The type is defined according to the OSA/Parlay standard. <code>P_AUTHENTICATION</code> is the only supported.</p> <p><b>Note:</b> When <code>P_AUTHENTICATION</code> is used, no encryption or signing algorithm will be used and the parameters <code>encryptionMethod</code> and <code>signingAlgorithm</code> can be left empty.</p>
encryptionMethod	<p>Method used for encryption. The type is defined according to OSA/Parlay standard. If the type is not specified, enter <code>P_RSA_1024</code>.</p>

**Table 30-6 AddMapping**

<b>addMapping</b>	
<b>Parameter</b>	<b>Description</b>
signingAlgorithm	Signing algorithm. The type is defined according to OSA/Parlay standard. If the type is not specified, enter P_MD5_RSA_1024.
gatewayId	OSA/Parlay Gateway ID. This ID was generated when the OSA/Parlay Gateway was created, see <a href="#">Operation: addGw</a> , and <a href="#">Operation: listGw</a> .
initConnection	Indicating if the connection to OSA/Parlay Gateway should be initialized immediately. That is, if authentication should performed when the <a href="#">Operation: addClient</a> operation is invoked.

## Operation: listActiveMappings

Scope: Cluster

Lists the IDs for active OSA/Parlay Client Mappings.

Signature:

```
listActiveMappings()
```

Returns a list of IDs for active mappings.

**Table 30-7 listActiveMappings**

<b>listActiveMappings</b>	
<b>Parameter</b>	<b>Description</b>
-	-

## Operation: listActiveMappingsForGw

Scope: Cluster

Lists the IDs of all active OSA/Parlay Client Mappings for a specific OSA/Parlay Gateway.

Signature:

```
listActiveMappingsForGw(gwId: int)
```

Returns a list of IDs for active mappings for the Gateway.

**Table 30-8 listActiveMappingsForGw**

<b>listActiveMappingsForGw</b>	
<b>Parameter</b>	<b>Description</b>
gwId	The ID of the OSA Gateway.

## Operation: listGw

Scope: Cluster

Lists the IDs of all registered OSA/Parlay Gateways.

Signature:

```
listGw()
```

**Table 30-9 listGw**

<b>listGw</b>	
<b>Parameter</b>	<b>Description</b>
-	-

## Operation: listMappings

Scope: Cluster

Lists the configured OSA/Parlay Client Mappings.

Signature:

```
listMappings()
```

**Table 30-10 listMappings**

listMappings	
Parameter	Description
-	-

## Operation: removeClient

Scope: Cluster

Removes an OSA/Parlay client.

Signature:

```
removeClient(osaClientAppId: String, keystorePwd: String)
```

**Table 30-11 removeClient**

removeClient	
Parameter	Description
osaClientAppId	The OSA/Parlay client application ID (and alias in keystore). See <a href="#">Operation: addClient</a> .
keystorePwd	Network Gatekeeper keystore password.

## Operation: removeConnection

Scope: Cluster

Removes an OSA/Parlay Gateway Connection

Signature:

```
removeConnection(gatewayId: int, connectionId: int)
```

**Table 30-12 removeConnection**

<b>removeConnection</b>	
<b>Parameter</b>	<b>Description</b>
gatewayId	The ID of the OSA/Parlay Gateway.
connectionId	The ID of the connection. The ID was returned when the connection was setup, see: <ul style="list-style-type: none"> <li>• <a href="#">Operation: addConnection</a></li> <li>• <a href="#">Operation: listActiveMappings</a></li> <li>• <a href="#">Operation: listMappings</a></li> </ul>

## Operation: removeGw

Scope: Cluster

Removes an OSA/Parlay Gateway.

Signature:

```
removeGw(id: int, keystorePwd: String)
```

**Table 30-13 removeGw**

<b>removeGw</b>	
<b>Parameter</b>	<b>Description</b>
id	The ID of the OSA/Parlay Gateway to remove. The ID was returned when the OSA Gateway was created, see <ul style="list-style-type: none"> <li>• <a href="#">Operation: addGw</a></li> <li>• <a href="#">Operation: listGw</a></li> </ul>
keystorePwd	The Network Gatekeeper keystore password.

## Operation: removeMapping

Scope: Cluster

Removes an OSA/Parlay client mapping.

Signature:

```
removeMapping(id: int)
```

Parameters:

**Table 30-14** removeMapping

removeMapping	
Parameter	Description
id	ID of the OSA/Parlay Client Mapping to remove.

## Operation: setKeyStorePassword

Scope: Cluster

Sets the password that protects the keystore.

Signature:

```
setKeyStorePassword(newPassword: String, oldPassword: String)
```

**Table 30-15** setKeyStorePassword

setKeyStorePassword	
Parameter	Description
newPassword	The new password for the keystore.
oldPassword	The old password for the keystore.

## Operation: viewActiveMappingState

Scope: Cluster

Displays the state of an active mapping OSA/Parlay Client Mapping.

Signature:

```
viewActiveMappingState(mappingId: int)
```



**Table 30-16 viewActiveMappingState**

<b>viewActiveMappingState</b>	
<b>Parameter</b>	<b>Description</b>
mappingId	The ID of the OSA Client Mapping.

## Managing OSA/Parlay Gateway Connections using Parlay\_Access