Oracle® Communications Services Gatekeeper

System Administrator's Guide

Release 4.1

September 2009

ORACLE

Oracle Communication Services Gatekeeper System Administrator’s Guide, Release 4.1

Contents

7. Managing and Configuring EDRs, CDRs and Alarms

8. CDRs and Diameter

9. Managing and Configuring Credit Control Interceptors and SLAs

10. Managing and Configuring Statistics and Transaction Licenses

11.Setting Up Geographic Redundancy

12.Managing and Configuring the SNMP service

13.Managing and Configuring the Trace Service

14.Managing and Configuring the Storage Service

15.Managing the Policy Service

16.Setting up WS-Policy and JMX Policy

17.Deployment Model for Communication Services and Container Services

18.Hitless Upgrade Using Production Redeployment

19.Managing and Configuring the Plug-in Manager

20.Managing and Configuring the Tier Routing Manager

21.Managing and Configuring Third Party Call Communication Services

22.Managing and Configuring Call Notification Communication Services

23.Managing and Configuring Short Messaging Communication Services

27.Managing and Configuring Audio Call Communication Services

28.Managing and Configuring the Presence Communication Services

29.Managing and Configuring Subscriber Profile Communication Services

33.SOAP2SOAP Communication Services

34.Configuring Heartbeats

35.Managing and configuring Shortcode mappings

36.Managing OSA/Parlay Gateway Connections using Parlay_Access

37.Resolving Policy Deny Codes

CHAPTER 1

Introduction

This guide describes Oracle Communications Services Gatekeeper from 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
- CDRs and Diameter
- 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 and Configuring Credit Control Interceptors and SLAs
- 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 the Tier Routing 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 Payment 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
- Managing OSA/Parlay Gateway Connections using Parlay_Access
- Managing and Configuring SOA Facades

Overall Configuration Workflow

This section describes an overall workflow for configuring Oracle Communications Services Gatekeeper.

1. To setup administrative users: Managing Management Users and Management User Groups.
2. To configure Oracle Communications Services 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
- Managing and Configuring the Tier Routing Manager
- Setting Up Geographic Redundancy

3. To configure Oracle Communications Services 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.

6. To configure SOA Facades (as necessary):
   - Managing and Configuring SOA Facades.

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*.

CHAPTER 2

Starting and Stopping Servers

The following sections describe how to start and stop servers in an Oracle Communications Services Gatekeeper domain:

Note: Oracle Communications Services 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 Oracle Communications Services 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.

Note: When OCSG is first used, the database is initialized. If two NT servers are started up at the same time, they may both try to initialize the database, causing a Service Deployment

Exception and one server failing to start up. Make sure the servers are started one at a time. If this occurs, restart the failed server.

3. Start Access Tier servers in each partition.

Caution: 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* http://download.oracle.com/docs/cd/E12840_01/wls/docs103/server_start/overview.html for more information.

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 *Oracle Weblogic Server Node Manager Administrator's Guide* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/nodemgr/index.html) 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>/ocsg_4.1/wlserver_10.3/server/bin` directory. Run the `./startNodeManager.sh` script.

2. Edit the `<bea_home>/wlserver_10.3/common/nodemanager/nodemanager.domains` file. This file specifies the domains that a Node Manager instance controls. See *Oracle*

Weblogic Server Node Manager Administrator's Guide at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/nodemgr/index.html for a description of `nodemanager.domains`. A sample entry: `ocsg-domain=/bea/user_projects/domains/ocsg-domain`

3. Edit the `<bea_home>/wlserver_10.3/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 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 Oracle Communications Services 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 a quick reference on how to start and stop servers, see *Oracle Weblogic Server Managing Server Startup and Shutdown* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/server_start/index.html.

Note: If you are using HP-UX in your installation, you must make a small edit to whichever of the startup scripts you are using. You must add the `-Djava.security.egd` flag to the invocation, in the locations shown below in bold.

```
if [ '${WLS_REDIRECT_LOG}' = '' ] ; then

        echo 'Starting WLS with line:'

        echo '${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS}
${JAVA_OPTIONS}-Dweblogic.Name=${SERVER_NAME}
-Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy
${PROXY_SETTINGS}

${SERVER_CLASS}' ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS}
${JAVA_OPTIONS} -Djava.security.egd=/dev/random -Dweblogic.Name
```

```
=${SERVER_NAME}
-Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy

${PROXY_SETTINGS} ${SERVER_CLASS}

else

        echo 'Redirecting output from WLS window to ${WLS_REDIRECT_LOG}'

        ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}

-Djava.security.egd=/dev/random -Dweblogic.Name

=${SERVER_NAME}
-Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy

${PROXY_SETTINGS} ${SERVER_CLASS}  >'${

WLS_REDIRECT_LOG}' 2>&1
```

Note: If your installation is for Solaris 64-bit, and you are using Sun's JVM, you must add the `-d64` flag to whichever startup script you are using. If you do not use this flag, the JVM will default to 32-bit.

CHAPTER 3

Operation and Maintenance: General

The following sections give an overview of the mechanisms available for controlling Oracle Communications Services Gatekeeper.

- Administration Console Overview
  - Administration Console
    - MBeans pane
    - Configuration and Provisioning
    - WebLogic Oracle Communications Services Gatekeeper Alarms pane
    - Oracle Communications Services Gatekeeper CDRs Pane
  - Oracle Communications Converged Application Server Management Console for SIP-based Communication Services
- Java Management Extensions (JMX)
- WebLogic Scripting Tool (WSLT)
  - Working in Interactive Mode
    - Starting WLST and connecting to Oracle Communications Services Gatekeeper
    - Exiting WLST
    - Changing an attribute
  - Invoking an operation
  - Scripting WLST

Administration Console Overview

Oracle Communications Services Gatekeeper Administration Console provides a graphical user interface for configuring, managing, and provisioning Oracle Communications Services 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. Oracle Communications Services Gatekeeper servers started after logging in to the Administration Console are not displayed in the Administration 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 Oracle Communications Services Gatekeeper is started, use the username `weblogic` and password `weblogic`, which is the recommended value for the installation and initial domain configuration stage. 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 *Oracle WebLogic Server Securing WebLogic Resources Using Roles and Policies* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/secwlres/index.html.

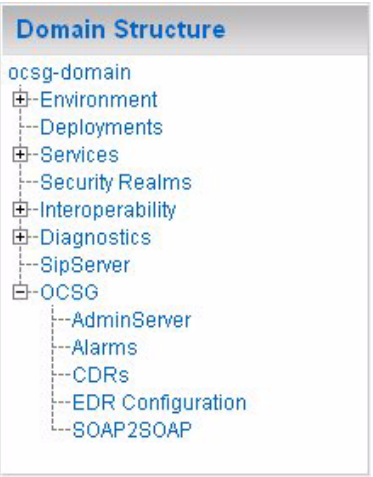
All Oracle Communications Services Gatekeeper configuration and monitoring is provided via the following nodes in the left pane of the console, in the **Domain Structure** group:

- OCSG — container for all Oracle Communications Services Gatekeeper servers
- <Server Name> — one entry per Oracle Communications Services Gatekeeper server.
  - <Server Name> — Clicking on this link displays the MBeans pane which displays all OAM objects available for the selected Oracle Communications Services Gatekeeper server. Some configuration settings are cluster-wide while other settings are per server.
  - Alarms — clicking this link displays the WebLogic Oracle Communications Services Gatekeeper Alarms pane.
  - CDRs — clicking this link displays the Oracle Communications Services 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.

- SipServer — clicking this link displays the Oracle Communications Converged Application Server's administration console, see Oracle Communications Converged Application Server Management Console for SIP-based Communication Services.

Figure 3-1 Domain Structure -links to management console for Oracle Communications Services Gatekeeper and Oracle Communications Converged Application Server



MBeans pane

The MBeans pane contains a tree-structure of all management objects applicable for the selected Oracle Communications Services Gatekeeper server.

By clicking on a management object, the corresponding Configuration and Provisioning page for the management object is displayed immediately below the MBean pane: see Configuration and Provisioning.

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.

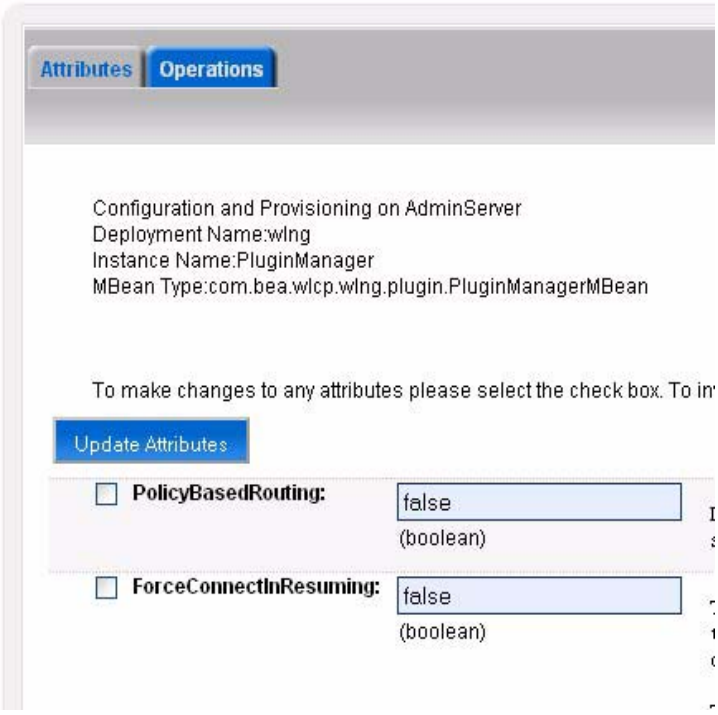
Note: In this document, read-only attributes are indicated by **(r)** after the name.

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 **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.

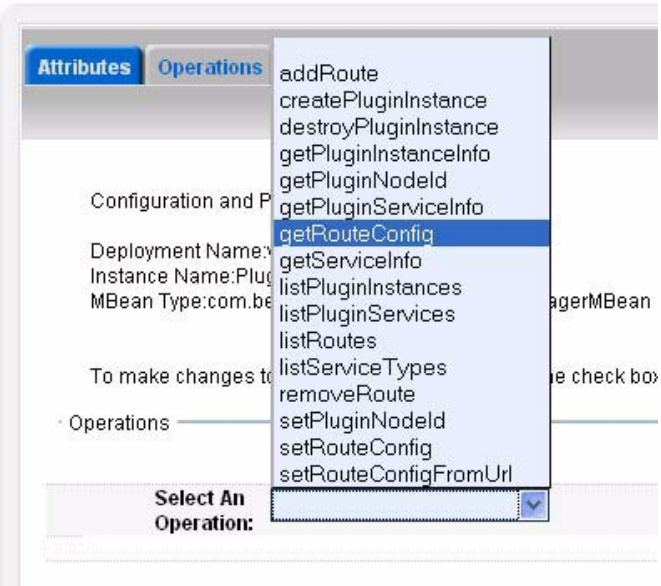
To perform an operation:

1. Select the operation from the drop-down-list **Select An Operation**.

   The entry-fields for the operation are displayed.

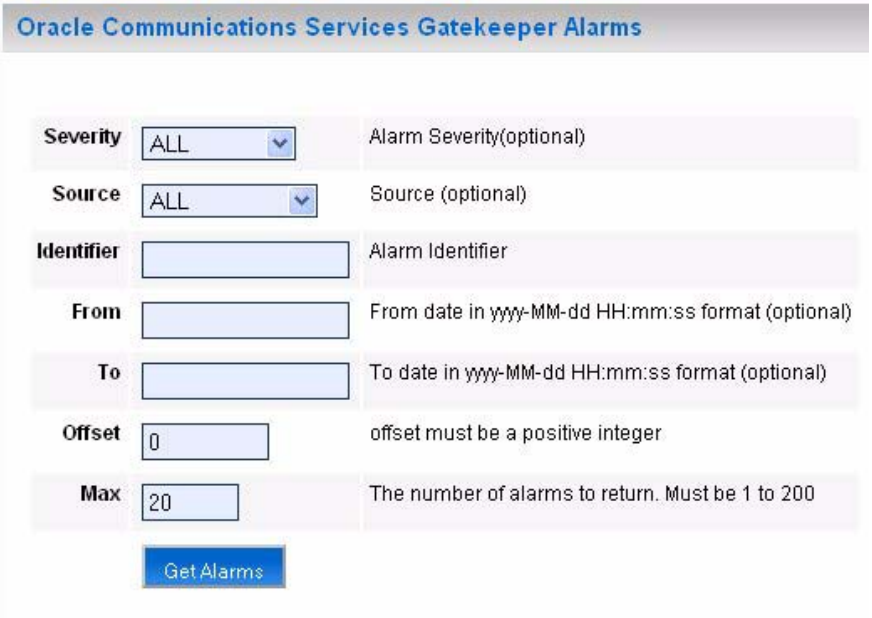
2. Enter the information in the entry fields.
3. Click **Invoke** to perform the operation.

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



WebLogic Oracle Communications Services Gatekeeper Alarms pane

Figure 3-5 Alarms pane



The Oracle Communications Services 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 Oracle Communications Services Gatekeeper Alarms pane

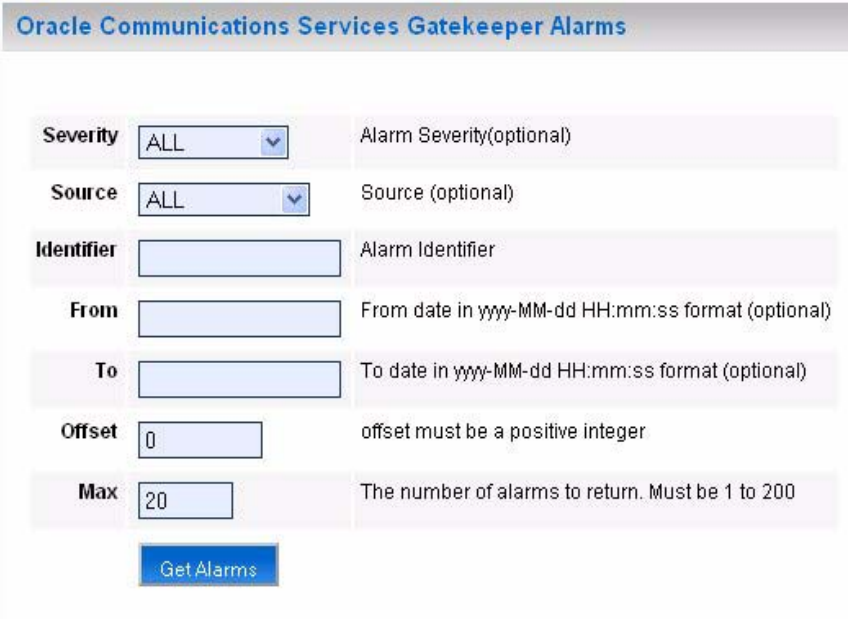
Filter on...	Input
Severity of alarms	From the drop-down list labeled **Severity**, choose which severity level to display: The options are: • **ALL** • **WARNING** • **MINOR** • **MAJOR** • **CRITICAL**
The server from which the alarm originates	From the drop-down list labeled **Server**, choose the server from which to display alarms. The options are: • **ALL**, for all servers • <**Server name**>, for an individual server.
ID of the alarm.	See *Oracle Communications Services Gatekeeper Handling Alarms* for alarm codes. Use 0 (zero) to wildcard.
Time interval	In the field labeled **From**, enter the start time for the time interval. The options are: • Exact time. • No given start time. Leave empty. In the field labeled **To**, enter the end time for the time interval. The options are: • Exact time. • No given time. Leave empty. Exact times are formatted as YYYY-MM-DD hh:mm., where: • YYYY is the year. Four digits required. • MM is the month [1...12] • DD is the day [1...[28|29|29|30|31]]. Upper limit depends on month and year. • hh is the hour [0...23] • mm is the minute [0...59] **Note:** There must be a white space separating YYYY-MM-DD and hh:mm.

Table 3-1 Oracle Communications Services Gatekeeper Alarms pane

Filter on...	Input
Offset	In the field labeled **Offset**, enter the start offset in the list of alarms entries matching the criteria. Integer. 0 (zero) is the first entry.
Max	In the field labeled **Max**, enter the maximum number of alarm entries to return. Integer.

Oracle Communications Services Gatekeeper CDRs Pane

Figure 3-6 CDRs pane



The Oracle Communications Services 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 Oracle Communications Services 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: • Exact time. • No given start time. Leave empty. Exact time is formatted as YYYY-MM-DD hh:mm:ss., where: • YYYY is the year. Four digits required. • MM is the month [1...12] • DD is the day [1...[28|29|29|30|31]]. Upper limit depends on month and year. • hh is the hour [0...23] • mm is the minute [0...59] • ss is the seconds [0...59] **Note:** There must be a white space separating YYYY-MM-DD and hh:mm:ss.

Table 3-2 Oracle Communications Services Gatekeeper CDRs pane

Filter on...	Input
To	Enter the end time for the time interval. The options are: • Exact time. • No given time. Leave empty. Exact times are formatted as YYYY-MM-DD hh:mm:ss., where: • YYYY is the year. Four digits required. • MM is the month [1...12] • DD is the day [1...[28|29|29|30|31]]. Upper limit depends on month and year. • hh is the hour [0...23] • mm is the minute [0...59] • ss is the seconds [0...59] **Note:** There must be a white space separating YYYY-MM-DD and hh:mm:ss.
Offset	Enter the start offset in the list of alarms entries matching the criteria. Integer. 0 (zero) is the first entry.
Max	Enter the maximum number of alarm entries to return. Integer.

Oracle Communications Converged Application Server Management Console for SIP-based Communication Services

There is an administration console for the Oracle Communications Converged Application Server. This handles setting for the Oracle Communications Converged Application Server and the SIP Servlet parts of Oracle Communications Services Gatekeeper communication services.

Click <Domain name>–>SipServer to open the Oracle Communications Converged Application Server administration console.

Java Management Extensions (JMX)

Oracle Communications Services 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 Oracle Communications Services Gatekeeper documentation set. See the reference section for each OAM service for the fully qualified MBean names and how they map to managed objects in Oracle Communications Services Gatekeeper Administration Console

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

The MBean object name is versioned. The MBean name includes the version number for the release. External JMX clients needs to be updated according to the release number.

Listing 3-1 Example of using JMX to manage Oracle Communications Services 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();

    h.put(Context.SECURITY_PRINCIPAL, username);

    h.put(Context.SECURITY_CREDENTIALS, password);

    h.put(JMXConnectorFactory.PROTOCOL_PROVIDER_PACKAGES,

        "weblogic.management.remote");

    connector = JMXConnectorFactory.connect(serviceURL, h);

    connection = connector.getMBeanServerConnection();
```

```
}

public static void main(String[] args) throws Exception {

  String hostname = args[0];   //hostname of the admin server

  String portString = args[1]; //port of the admin server

  String username = args[2];

  String password = args[3];

  String mbserverName = args[4];  //NT server name

  String operationName = "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.wlng:Name=wlng,InstanceName=PluginManager,Type=
com.bea.wlcp.wlng.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.wlng.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/ocsg_pds4.1/lib/wlng/oam.jar` contains custom classes for OAM and return types.

`$BEA_Home/ocsg_pds4.1/doc/javadoc_oam` contains JavDoc for OAM.

WebLogic Scripting Tool (WSLT)

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

For information about WebLogic Server and WLST, see *Oracle WebLogic Server WebLogic Scripting Tool* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/config_scripting/index.html.

The following topics are covered:

Note: A link to JavaDoc for OAM can be found on the *Reference* topic page in the Oracle Communications Services Gatekeeper documentation set.

Working in Interactive Mode

Starting WLST and connecting to Oracle Communications Services Gatekeeper

1. Make sure the correct Java environment is set:

   UNIX: `$Domain_Home/bin/setDomainEnv.sh`

   or

   Windows: `$Domain_Home\bin\setDomainEnv.cmd`

2. Start WLST: `java weblogic.WLST`

3. Connect to the server to manage:

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

4. Change to the custom tree where Oracle Communications Services 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 Oracle Communications Services Gatekeeper Management Console.

6. Select the MBean to change:

   ```
   cd('com.bea.wlcp.wlng:Name=wlng_nt,Type=<MBean name>')
   ```

   For example, to select the MBean for the EDRService, use:

   ```
   cd('com.bea.wlcp.wlng:Name=wlng,InstanceName=EdrService,Type=com.bea.wl
   cp.wlng.edr.management.EdrServiceMBean')
   ```

Exiting WLST

1. Disconnect from Oracle Communications Services 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.wlng:Name=wlng,InstanceName=EdrService,Type=com.bea.wlcp.wlng.edr.management.EdrServiceMBean')

objs = jarray.array([],java.lang.Object)

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

print(invoke('displayStatistics',objs,strs))
```

This operation has no arguments.

Another example: to add a route using the MBean for the Plug-in Manager:

```
cd(‘com.bea.wlcp.wlng:Name=wlng,InstanceName=PluginManager,Type=com.bea.wlcp.wlng.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('stringInp
ut2'), 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 Oracle Communications Services 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.wlng:Name=nt,Type=" + sys.argv[5]

print objectName

connect(userName, passWord, url)

custom()

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

cd(objectName)
```

For example: To invoke the script:

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

CHAPTER 4

Managing Management Users and Management User Groups

The following section describes how to set up and manage administrative users of Oracle Communications Services Gatekeeper.

Overview

Management of Oracle Communications Services 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	Operation: addUser
List administrative users	Operation: listUsers
Change password	Operation: changeUserPassword
Delete an administrative user	Operation: deleteUser

Users and User Groups

Oracle Communications Services 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 • They should be able to just send traffic and should not have access to management functions. • They should not have access to WebLogic Server or Oracle Communications Services Gatekeeper MBeans. • They should not be able to log into the console and perform WebLogic Server administration operations.	TrafficUser
OamUser	Management users who are of OAM type: • They have access to the console based on their level. • They should not be able to send traffic.	OamUser
PrmUser	Management users who are of type PRM: • They should not have access to the console. • They should perform their management operations using the PRM interfaces	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 Oracle Communications Services Gatekeeper.
- **PRM operator users** use the PRM Operator Web Services interfaces to interact with Oracle Communications Services Gatekeeper.
- **PRM service provider users** use the PRM Service Provider Web Services interfaces to interact with Oracle Communications Services 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 Oracle Communications Services Gatekeeper	Access on WebLogic Server
1000	Administration access to management functions.	Administration access: • View, modify and administer server configuration • Deploy applications • Start, resume and stop servers
666	Read-write access on management functions.	Deployer access: • View the server configuration, including some encrypted attributes related to deployment activities. • Change startup and shutdown classes, Web applications, JDBC data pool connections, EJB, Java EE Connector, Web Service. If applicable, edit deployment descriptors. • Access deployment operations in the Java EE Deployment Implementation (JSR-88).
333	Read-only access on management functions.	Monitor access: • View the server configuration • Have read-only access to Administration Console, WLST and other MBean APIs
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 *Oracle WebLogic Server Securing WebLogic Resources Using Roles and Policies* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/secwlres/index.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.wlng.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 an Oracle Communications Services Gatekeeper administrative user.

Signature:

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

Table 4-2 addUser

addUser	
Parameter	**Description**
Username	User name.
Password	Password
UserLevel	Defines the user level when administrating Oracle Communications Services Gatekeeper. See User Level.
Type	Type of management user. Use: 0 for management user 1 for PRM operator user 2 for PRM service provider user See User Types.

Operation: changeUserPassword

Scope: Cluster

Changes the password for an existing Oracle Communications Services Gatekeeper administrative user.

Signature:

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

Table 4-3 changeUserPassword

changeUserPassword	
Parameter	**Description**
UserName	User ID for administrative user.
OldPasswd	Current password.
NewPasswd	New password.

Operation: deleteUser

Scope: Cluster

Deletes an Oracle Communications Services 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 User Types.
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.wlng.user.management.ManagementUserGroupMBean

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

Operation: addUsertoGroup

Scope: Cluster

Adds an Oracle Communications Services Gatekeeper administrative user to a user group.

Signature:

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

Table 4-7 addUsertoGroup

addUsertoGroup	
Parameter	**Description**
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

createUserGroup	
Parameter	**Description**
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

listGroups	
Parameter	**Description**
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

deleteUser	
Parameter	**Description**
GroupName	Group name.
Offset	Offset in the list. Starts with 0.
Size	Size of the list.

Managing and Configuring SOA Facades

The following sections describe how to configure the Oracle Communications Services Gatekeeper SOA Facades in Oracle Service Bus:

Introduction

All servers in a SOA deployment are Oracle Communications Services Gatekeeper servers with Oracle Service Bus server installed over them and configured using the SOA domain configuration template provided with Oracle Communications Services Gatekeeper. The SOA Facades are managed using the Oracle Service Bus management console.

The oracle Service bus administration console is located at:

```
http://<IP-address of Administration Server>/sbconsole
```

For each communication service there is a set of business services and proxy services, one for each SOAP interface.

There is no need to configure the SOA Facades, but both the business services and the proxy services can be configured in the same manner as other oracle Service Bus projects. The project for the SOA Facades needs to be loaded using the Service Bus Console.

Load SOA Facade Projects

This section describes how to load the SOA Facade projects.

The SOA Service Facades are not loaded by default. Load the projects for the Communication Services that are to be used using the instruction below.

See Available SOA Facades for a list of available SOA Facades and the deployment artifacts for them.

1. In the Service Bus administration console, click on **System Administration**.
2. Under **System Administration**, click on **Import Resources**.
3. Click the **Browse** button next to the **File Name** entry field.
4. Browse to the directory `$OCSG_DOMAIN_HOME/soa` and select the file, according to Table 5-1.
5. In the **Import Resources- Project JAR File** screen, click the **Import** button.

Note: In the **Change Center**, click the **Activate** button.

Available SOA Facades

Below is a list of SOA Facades for Oracle Communications Services Gatekeeper. The files are located in `$OCSG_DOMAIN_HOME/soa`.

Table 5-1 SOA Facade projects and services

Service Facade	Project	File
-	soa_common	File: sb_common.jar Business Service: n/a Proxy Service: n/a Type: Use for both application-initiated and network-triggered requests.
Session Manager	soa_session	File: sb_session.jar Business Service: SessionManager_BS Proxy Service: SessionManager_PS Type: Use for application-initiated requests.

Table 5-1 SOA Facade projects and services

Service Facade	Project	File
Parlay X 3.0 Audio Call	soa_audio_call_px30	File: sb_ac_px30.jar Business Service: AudioCallCaptureMedia_BS Business Service: AudioCallPlayMedia_BS Proxy Service: AudioCallCaptureMedia_PS Proxy Service: AudioCallPlayMedia_PS Type: Use for application-initiated requests.
Parlay X 2.1 Call Notification	soa_call_notification_px21	File: sb_cn_px21.jar Business Service: CallDirection_BS Business Service: CallDirectionManager_BS Business Service: CallNotification_BS Business Service: CallNotificationManager_BS Proxy Service: CallDirection_PS Proxy Service: CallDirectionManager_PS Proxy Service: CallNotification_PS Proxy Service: CallNotificationManager_PS Type: Use for network-triggered requests.
Parlay X 3.0 Call Notification	soa_call_notification_px30	File: sb_cn_px30.jar Business Service: CallDirection_BS Business Service: CallDirectionManager_BS Business Service: CallNotification_BS Business Service: CallNotificationManager_BS Proxy Service: CallDirection_PS Proxy Service: CallDirectionManager_PS Proxy Service: CallNotification_PS Proxy Service: CallNotificationManager_PS Type: Use for network-triggered requests.

Table 5-1 SOA Facade projects and services

Service Facade	Project	File
Callable Policy	soa_callable_policy	File: sb_callable_policy.jar Business Service: Policy_BS Business Service: PolicyManagement_BS Proxy Service: Policy_PS Proxy Service: PolicyManagement_PS Type: Use for application-initiated requests.
Parlay X 2.1 Multimedia Messaging	soa_mms_px21	File: sb_mms_px21.jar Business Service: MessageNotification_BS Business Service: MessageNotificationManager_BS Business Service: ReceiveMessage_BS Business Service: SendMessage_BS Proxy Service: MessageNotification_PS Proxy Service: MessageNotificationManager_PS Proxy Service: ReceiveMessage_PS Proxy Service: SendMessage_PS Type: Use for both application-initiated and network-triggered requests.
Parlay X 3.0 Payment	soa_payment_px30	File: sb_payment_px30.jar Business Service: AmountCharging_BS Business Service: ReserveAmountCharging_BS Proxy Service: AmountCharging_PS Proxy Service: ReserveAmountCharging_PS Type: Use for application-initiated requests.

Table 5-1 SOA Facade projects and services

Service Facade	Project	File
Parlay X 2.1 Presence	soa_presence_px21	File: sb_presence_px21.jar Business Service: PresenceConsumer_BS Business Service: PresenceNotification_BS Business Service: PresenceSupplier_BS Proxy Service: PresenceConsumer_PS Proxy Service: PresenceNotification_PS Proxy Service: PresenceSupplier_PS Type: Use for both application-initiated and network-triggered requests.
Parlay X 2.1 Short Messaging	soa_sms_px21	File: sb_sms_px21.jar Business Service: ReceiveSms_BS Business Service: SendSms_BS Business Service: SmsNotification_BS Business Service: SmsNotificationManager_BS Proxy Service: ReceiveSms_PS Proxy Service: SendSms_PS Proxy Service: SmsNotification_PS Proxy Service: SmsNotificationManager_PS Type: Use for both application-initiated and network-triggered requests.
Extended Web Services Subscriber Profile	soa_subscriber_ews	File: sb_subscriber_ews.jar Business Service: SubscriberProfile_BS Proxy Service: SubscriberProfile_PS Type: Use for application-initiated requests.
Parlay X 2.1 Third Party Call	soa_third_party_call_px21	File: sb_tpc_px21.jar Business Service: ThirdPartyCall_BS Proxy Service: ThirdPartyCall_PS Type: Use for application-initiated requests.

Table 5-1 SOA Facade projects and services

Service Facade	Project	File
Parlay X 3.0 Third Party Call	soa_third_party_call_px30	File: sb_tpc_px30.jar Business Service: ThirdPartyCall_BS Proxy Service: ThirdPartyCall_PS Type: Use for application-initiated requests.
Parlay X 2.1 Terminal Location	soa_tl_px21	File: sb_tl_px21.jar Business Service: TerminalLocation_BS Business Service: TerminalLocationNotification_BS Business Service: TerminalLocationNotificationManager_BS Proxy Service: TerminalLocation_PS Proxy Service: TerminalLocationNotification_PS Proxy Service: TerminalLocationNotificationManager_PS Type: Use for both application-initiated and network-triggered requests.

Table 5-1 SOA Facade projects and services

Service Facade	Project	File
Extended Web Services WAP Push	soa_wap_ews	File: sb_wap_ews.jar Business Service: PushMessage_BS Business Service: PushMessageNotification_BS Proxy Service: PushMessage_PS Proxy Service: PushMessageNotification_PS Type: Use for application-initiated requests.
Extended Web Services Binary SMS	soa_sms_ews	File: sb_sms_ews.jar Business Service: BinarySms_BS Business Service: BinarySmsNotification_BS Business Service: BinarySmsNotificationManager_BS Proxy Service: BinarySms_PS Proxy Service: BinarySmsNotification_PS Proxy Service: BinarySmsNotificationManager_PS Type: Use for both application-initiated and network-triggered requests.

CHAPTER 6

Managing and Configuring Budgets

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

Introduction

In Oracle Communications Services 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 Oracle Communications Services Gatekeeper server updates both its own local traffic count and the cluster-wide count maintained in one Oracle Communications Services 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 its state from the persistent store. If a cluster master fails, each Oracle Communications Services 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 persisting data has on overall performance, and the 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), 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_t 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 Tt = 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 override is active. If an override is active, the budgt master enforces limits based on that 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.

Budget Calculations and Relationship to SLA Settings

Budgets are calculated based on the following SLA settings:

- <reqLimit> and <timePeriod> for request limits.
- <qtaLimit> and <days> for quotas.
- <reqLimitGuarantee> and <timePeriodGuarantee> for guarantee settings

The limits divided by the time-period translates into a budget increase rate, expressed in transactions per second.

Each budget has a maximum value define in the limits (<reqLimit>, <qtaLimit, and <reqLimitGuarantee>). This value is also the starting value of the budget. For each requests that is processed, the budget is decreased with one (1). Over time, the budget increases with the budget increase rate multiplied with the time, and its maximum value is the request limit. When a budget has reached the value of zero (0), requests are denied.

A maximum request rate is expressed as the number of request during a a time-period, so it offers very flexible ways to define the limits. For a given budget increase rate, expressed in transactions per second:

- The longer time-period defined, the longer it takes for requests to start to be rejected if the request rate is higher than allowed since the maximum budget value is higher.
- The shorter time-period defined, the sooner requests are starting to be rejected if the request rate is higher than allowed since the maximum budget value is lower.

Having a shorter time period means that budget synchronizations are more frequent and this has a performance impact.

If an application has used up its budget by sending more requests than allowed, and requests have started to be rejected, and the application reduces the request rate, the budget starts to increase. The budget is increased with the delta between the budget increase rate and the request rate.

Example:

An application sends 250 requests per second.

The SLA settings are:

<reqLimit>2000</reqLimit>

<timePeriod>10000</timePeriod>

This means that the budget increase rate is 2000/10000= 200 requests per second.

The difference between the request rate and the budget increase factor is 50 (250-200) so the budget will be zero (0) after 40 seconds (2000/50). When the budget is zero, requests are rejected.

Example:

An application sends 250 requests per second.

The SLA settings are

<reqLimit>200</reqLimit>

<timePeriod>1000</timePeriod>

This means that the budget increase rate is 200/1000= 200 requests per second.

The difference between the request rate and the budget increase rate is 50 (250-200) so the budget will be zero (0) after 4 seconds (200/50). When the budget is zero, requests are rejected.

Example:

An application sends 180 requests per second.

The SLA settings are

<reqLimit>200</reqLimit>

<timePeriod>1000</timePeriod>

Also, the current budget is zero (0) since it previously had a request rate that was higher than the allowed.

This means that the budget increase rate is 200/1000= 200 requests per second.

The difference between the budget increase rate and the request rate is 20 (200-180) so the budget will be at its maximum value (200) after 10 seconds (200/20). When the budget reaches its maximum value it does not increase any further.

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.wlng.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, Oracle Communications Services 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 Oracle Communications Services 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 automatic migration of singleton services, see *Oracle WebLogic Using Clusters* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/cluster. Also see the section about high-availability database leasing in that document the mechanism underlying migration.

For information on setting up a separate datasource to support migration of singleton services, like the Budget service, see the section *Configuring JDBC Data Sources* in *Oracle WebLogic Server Configuring and Managing WebLogic JDBC* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/jdbc_admin/jdbc_datasources.html.

CHAPTER 7

Managing and Configuring EDRs, CDRs and Alarms

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

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, Oracle Communications Services 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 7-1 illustrates the structure of wlng-edr.xml. See section in *Oracle Communications Services Gatekeeper Platform Development Studio - Developer's Guide* for more information.

Listing 7-1 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>
```

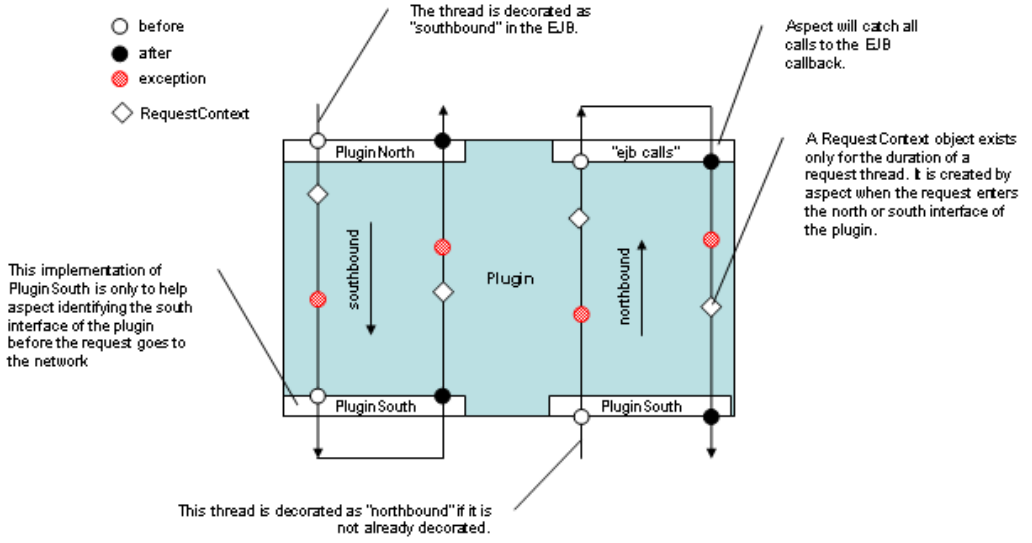
EDR format

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 Oracle Communications Services Gatekeeper to the network (south) or from the network to Oracle Communications Services 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 7-1 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 7-1 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 7-1 Contents of an EDR

String value of name (key) in name/ value pair	Description
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 section *in Oracle Communications Services Gatekeeper Platform Development Studio - Developer's Guide* 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

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 7-2 displayStatistics

displayStatistics	
Parameter	**Description**
-	-

Operation: resetStatistics

Scope: Cluster

Resets the statistics for the EDRService.

Signature:

```
resetStatistics()
```

Table 7-3 resetStatistics

resetStatistics	
Parameter	**Description**
-	-

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

The Oracle Communications Services Gatekeeper EDR Configuration pane allows the administrator to load new EDR, CDR, and Alarm configuration files.

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

Note: Lock and Edit must be used.

Table 7-4 Oracle Communications Services Gatekeeper EDR Configuration pane

Entry field	Input
EDR descriptor:	The EDR configuration file, see EDR categories and XML markup.
CDR descriptor	The CDR configuration file, see EDR categories and XML markup.
Alarm descriptor	The alarms configuration file, see EDR categories and XML markup.

CHAPTER 8

CDRs and Diameter

The following section describes how to manage and configure the CDR to Diameter service in Oracle Communications Services Gatekeeper.

About CDRs and Diameter

The CdrToDiameter service forwards charging events to a Diameter server using the DIAMTER Rf interface. This allows for easy integration with charging systems that support Diameter Rf offline charging. When enabling this service, Oracle Communications Billing and Revenue Management, together with Oracle Communications Network Mediation, can be used out-of-the-box with Oracle Communications Services Gatekeeper.

Any traffic request processed through Oracle Communications Services Gatekeeper that generates a CDR also generates a corresponding Diameter Rf request. The Diameter server can the be used for any billing, charging, and reporting for these requests. See CDR to AVP mapping for information about how CDRs are mapped to Diameter attribute-value pairs (AVPs).

CdrToDiameter Service Deployment Characteristics

The service is not deployed by default. It is deployed as a regular JEE module using WebLogic Server deployment tools. See *Oracle WebLogic Server Deploying Applications to WebLogic Server* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/deployment/ for information on how to deploy the service.

The service is packaged in `cdr_to_diameter-single.ear` and `cdr_to_diameter.ear` in `$OCSG_HOME/applications`.

Use `cdr_to_diameter.ear` in clustered installations, and `cdr_to_diameter-single.ear` in single server domains.

The service is a cluster singleton, so it will execute only on one server at any given time, and is transferred to another server in case of server failure.

The management part is distributed to all servers in the cluster, so it can be managed from any server in the cluster.

Note: Some Diameter requests may be dropped during patching, redeployment, or upgrade of the CDR to Diameter module. Check the database for the time period during which the transition took place. All CDRs are stored in the database.

Configuration of CdrToDiameter

To configure the behavior of the CdrToDiameter service, in the managed object CdrToDiameter:

1. Specify:
   - Attribute: OriginHost
   - Attribute: OriginPort
   - Attribute: DestinationHost
   - Attribute: DestinationPort
   - Attribute: DestinationRealm
   - Attribute: PeerRetryDelay
   - Attribute: RequestTimeout
   - Attribute: WatchdogTimeout
   - Attribute: OriginHost
2. Use Operation: connect to connect to the Diameter server.

Management of CdrToDiameter

The CdrToDiameter service can be explicitly connected to the Diameter server. It does not connect to the server by default. The service has a connection status that will be preserved after service redeployment and server restart.

Use:

- Operation: connect
- Operation: disconnect

.Use Attribute: Enabled (r) to check the current connection status.

Use Operation: connect after any changes to the configuration attributes. Changes does not take affect until this operation is invoked.

Reference: Attributes and Operations for CdrToDiameter

Managed object: Container Services–>CdrToDiameter

MBean: com.bea.wlcp.wlng.cdrdiameter.management.CDRDiameterMBean

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

- Attribute: Enabled (r)
- Attribute: OriginHost
- Attribute: OriginPort
- Attribute: DestinationHost
- Attribute: DestinationPort
- Attribute: DestinationRealm
- Attribute: PeerRetryDelay
- Attribute: RequestTimeout
- Attribute: WatchdogTimeout
- Operation: connect
- Operation: disconnect

Attribute: Enabled (r)

Read-only.

Scope: Cluster

Format: Boolean

Unit: n/a

Displays the status of the CdrToDiameter service.

Displays:

- `true`, if enabled.
- `false`, if not enabled.

Attribute: OriginHost

Scope: Server

Format: String

Unit: n/a

Specifies the Origin-Host AVP in the Diameter request.

Can be specified either as an IP-address or a host name.

Optional.

Attribute: OriginPort

Scope: Server

Format: int

Unit: n/a

Specifies the local originator port to be used for the connection to the Diameter server.

If specified as 0 (zero), a random local port is used.

Attribute: DestinationHost

Scope: Server

Format: String

Unit: n/a

Specifies the Destination-Host AVP in the Diameter request.

Can be specified either as an IP-address or a host name.

Attribute: DestinationPort

Format: int

Unit: n/a

Specifies the port on the Diameter server to connect to.

Attribute: DestinationRealm

Scope: Cluster

Format: String

Unit: n/a

Specifies the Destination-Realm AVP in the Diameter request.

Attribute: PeerRetryDelay

Scope: Cluster

Format: int

Unit: seconds

Specifies the time to wait before attempting to reconnect to the Diameter server if the connection is lost.

Attribute: RequestTimeout

Scope: Cluster

Format: int

Unit: milliseconds

Specifies the maximum time to wait for a response to a request to the Diameter server.

Attribute: WatchdogTimeout

Scope: Cluster

Format: int

Unit: seconds

Specifies the watchdog timeout Tw.

This setting corresponds to the timer Tw, see RFC 3539 at http://www.ietf.org/rfc/rfc3539.txt.

It specifies the time to wait before attempting to reconnect to the Diameter server in the case of a lost connection.

Operation: connect

Scope: Cluster

Connects to the Diameter server.

Once connected, the service will try to reconnect to the Diameter server if the server is restarted or the service is redeployed.

Signature:

```
connect()
```

Table 8-1 connect

connect	
Parameter	**Description**
-	-

Operation: disconnect

Scope: Cluster

Disconnects from the Diameter server.

Once disconnected, the service will not try to reconnect to the DAMETER server if the server is restarted or the service is redeployed.

Signature:

```
disconnect()
```

Table 8-2 disconnect

disconnect	
Parameter	**Description**
-	-

CDR to AVP mapping

The CDRs that are generated are mapped to Diameter attribute-value pairs, AVPs, according to Table 8-3. Both standard and custom AVPs are used. When custom are used it is indicated in the table.

Table 8-3 CDR to Diameter AVP mappings

AVP	Source	Description
Session-Id	Auto generated.	Operation session identifier. Specification: RFC 3588 Example: nt1.ocsg.oracle.com;1214342798;0
Origin-Host	Configurable, see Attribute: OriginHost.	Realm of the Oracle Communications Services Gatekeeper domain. Addressed with the domain address of the corresponding public URI. Specification: RFC 3588 Example: nt1.ocsg.oracle.com

Table 8-3 CDR to Diameter AVP mappings

AVP	Source	Description
Origin-Realm	Auto generated.	Ream of the Oracle Communications Services Gatekeeper domain. Addressed with the domain address of the corresponding public URI. Specification: RFC 3588 Example: oracle.com
Destination-Realm	Configurable, see Attribute: DestinationRealm.	Realm of the operator domain. Addressed with the domain address of the corresponding public URI. Specification: RFC 3588 Example: oracle.com
Accounting-Record-Type	Value of CDR name-value pair: CallInfo	Defines the transfer type. Value is EVENT_RECORD for event based charging. Value is START_RECORD, INTERIM_RECORD, or STOP_RECORD for session based charging. If value of CallInfo is START, transfer type is EVENT_RECORD or START_RECORD. If value of CallInfo is STOP, transfer type is STOP_RECORD. EVENT_RECORD has numeric value 1. START_RECORD has numeric value 2. INTERIM_RECORD has numeric value 3. STOP_RECORD has numeric value 4. Specification: RFC 3588 Example: 1

Table 8-3 CDR to Diameter AVP mappings

AVP	Source	Description
Accounting-Record-Number	Auto generated.	Sequence number of the Diameter message sent from the CdrToDiameter service. Specification: RFC 3588 Example: 42
Acct-Application-Id	Static. Value is always 3.	Value is always 3, which corresponds to the application ID of the Diameter Accounting Application (off-line charging). Specification: RFC 3588
User-Name	Value of CDR name-value pair: ServiceProviderId	The service provider ID associated with the request that triggered the CDR. Specification: RFC 3588 Example: service_provider_1
Event-Timestamp	CDR attribute in name-value pair: Timestamp	Time the event happened. Given in seconds since the year 1900. Specification: RFC 3588
Calling-Party-Address	Value of CDR name-value pair: destinationParty	Custom AVP. Depending on which communication service that triggered the CDR, different application-provided parameters are used. Specification: 3GPP 32.299 Example: tel:7878

Table 8-3 CDR to Diameter AVP mappings

AVP	Source	Description
Called-Party-Address	Value of CDR name-value pair: originatingParty	Custom AVP. Depending on which communication service that triggered the CDR, different application-provided parameters are used. Specification: 3GPP 32.299 Example: tel:7878
Service-Indication	Value of CDR name-value pair: ServiceName	Custom AVP. The name of the service the request triggered. Specification: 3GPP 29.329 Example: MultimediaMessaging
Message-Size	Calculated from message payload. Relevant for Parlay X 2.1 Short Messaging and Multimedia Messaging.	Custom AVP. For MM, it holds the total size in bytes of the MM calculated according to TS 23.140 For SM, it holds the total size in octets of the SM including any user data header. Specification: 3GPP 32.299 Example: 345
OCSG-Charge-Description	Parameter ChargingDescription in any Parlay X operation that includes this parameter.	Custom AVP. Application-provided description text to be used for information and billing text. Specification: none Example: Delivery of weather report.

Table 8-3 CDR to Diameter AVP mappings

AVP	Source	Description
Currency-Code	Parameter ChargingCurrency in any Parlay X operation that includes this parameter.	Custom AVP. Currency code mapped according to ISO 4217. Specification: RFC 4006 Example: If the currency is U.S dollars, the value is 840
Unit-Value.Exponent	Parameter ChargingAmount in any Parlay X operation that includes this parameter.	Custom AVP. The exponent value to be applied for the Value-Digit AVP within the Unit-Value AVP. Specification: RFC 4006 Example: -2
Unit-Value.Value-Digits	Parameter ChargingAmount in any Parlay X operation that includes this parameter.	Custom AVP. The value to be applied for the Value-Digit AVP within the Unit-Value AVP. Contains the significant digits of the number scaled to an integer. Specification: RFC 4006 Example: 4062
Service-Context-Id	Parameter ChargingCode in any Parlay X operation that includes this parameter.	Custom AVP. Code that references a contract under which the charge is applied. Specification: RFC 4006 Example: premium

CHAPTER 9

Managing and Configuring Credit Control Interceptors and SLAs

The following section describes how to manage and configure the Credit Control Interceptors in Oracle Communications Services Gatekeeper.

Introduction

The Credit Control Interceptors performs credit checks towards a Diameter server using the Diameter Ro interface. This allows for easy integration with charging systems that support Diameter Ro on-line charging. Oracle Communications Billing and Revenue Management, can be used out-of-the-box with Oracle Communications Services Gatekeeper.

Traffic flowing through any Communication Service is intercepted and the data in the request, together with other configuration data, is passed on to a Diameter server for credit control verification. Both application-initiated and network-triggered requests are intercepted, there is one interceptor for each direction.

The credit control check is performed either synchronous or asynchronous.

Application-Initiated Requests

In the synchronous case, money is reserved before the request is passed on to the telecom network node. When the request has been successfully handed off and the thread of execution is returning, the reservation is committed. In case the requests fails, the reservation is cancelled.

In the asynchronous case, money is reserved before the request is passed on to the telecom network node. When a correlated request arrives to the interceptors, the reservation is committed. In case the correlated request does not arrive within a set period of time, the reservation is cancelled.

Network-Triggered Requests

Money is reserved before the request is passed on to the application. When the request has been successfully handed off and the thread of execution is returning, the reservation is committed. In case the requests fails, the reservation is cancelled.

Only synchronous reservations are supported.

Credit Control Interception Points

Only requests that are explicitly defined to be subject to credit control checks are passed on to the Diameter server.

It is possible to map a parameter in the request to be passed in the Diameter request, or to define a static value for the following AVPs:

- User-Name

- OCSG-Charge-Description
- Service-Context-Id
- Unit-Value.Exponent and Unit-Value.Value-Digits
- Currency-Code

The mapping is expressed in XML and provisioned as a service level agreement on service provider group or application group level.

See for information about how CDRs are mapped to Diameter attribute-value pairs (AVPs).

Writing Credit Control SLAs

Credit Control SLAs defines which methods that shall be subject to credit control, how the credit control shall be performed, and which data to provide to the Diameter server.

Table 9-1 Structure and contents of a credit control SLA

Tag	Description
<CCInterceptions>	Main tag. Defines a set of interception points for credit controls. Contains: <CCInterception>, one (1) or more
<CCInterception>	Defines how and when to trigger a credit control reservation. Used both in asynchronous and synchronous credit control checks Has the following attributes: • InterfaceName, which defines the Service Gatekeeper interface that the method defined in the attribute methodName belongs to. • methodName, which defines the name of the method that the credit control check applies to. See section Managing and Configuring the Plug-in Manager in *Oracle Communications Services Gatekeeper Administration Guide* for information on how to get a list of interface names and methods. For example, the interface name for Parlay X 2.1 Short Messaging and RESTful Short Messaging send SMS is com.bea.wlcp.wlng.px21.plugin.SendSmsPlugin and the method that sends the SMS is sendSms. Contains: <SubscriptionId>, exactly one (1). <OCSGChargeDescription>, zero (0) or one (1). <ServiceContextId>, zero (0) or one (1). <Amount>, zero (0) or one (1). <Currency>, zero (0) or one (1). <AsynchronousCommit>, zero (0) or one (1).

Table 9-1 Structure and contents of a credit control SLA

Tag	Description
<SubscriptionId>	Parent tag: <CCInterception>. Defines how the Diameter AVP User-Name shall be populated. The alternatives are to dynamically fetch the value from the request that is being subject to credit control or to pass in a configurable value. Has the following attributes: • type, that defines if the value of the AVP shall be fetched dynamically or statically. Use the keyword **Dynamic** to fetch the value from the request. Use the keyword **Static** to define a static value. • value, that defines what to send in the AVP. If the attribute type is set to **Dynamic**, use the fully qualified name of the parameter to use. If the attribute type is set to **Static**, enter the string to use in the AVP. See Defining Static and Dynamic Parameter Mappings.
<OCSGChargeDescription>	Parent tag: <CCInterception>. Defines how the Diameter AVP OCSG-Charge-Description shall be populated. The alternatives are to either dynamically fetch the value from the request that is being subject to credit control or to pass in a configurable value. Has the following attributes: • type, that defines if the value of the AVP shall be fetched dynamically or statically. Use the keyword **Dynamic** to fetch the value from the request. Use the keyword **Static** to define a static value. • value, that defines what to send in the AVP. If the attribute type is set to **Dynamic**, use the fully qualified name of the parameter to use, If the attribute type is set to **Static**, enter the string to use in the AVP. See Defining Static and Dynamic Parameter Mappings.

Table 9-1 Structure and contents of a credit control SLA

Tag	Description
<ServiceContextId >	Parent tag: <CCInterception>. Defines how the Diameter AVP Service-Context-Id shall be populated. The alternatives are to either dynamically fetch the value from the request that is being subject to credit control or to pass in a configurable value. Has the following attributes: • type, that defines if the value of the AVP shall be fetched dynamically or statically. Use the keyword **Dynamic** to fetch the value from the request. Use the keyword Static to define a static value. • value, that defines what to send in the AVP. If the attribute type is set to **Dynamic**, use the fully qualified name of the parameter to use. If the attribute type is set to **Static**, enter the string to use in the AVP. See Defining Static and Dynamic Parameter Mappings. If not present, the value is set to current time, given in milliseconds, from 1970.
<OCSGChargeDescription>	Parent tag: <CCInterception>. Defines how the Diameter AVP OCSG-Charge-Description shall be populated. The alternatives are to either dynamically fetch the value from the request that is being subject to credit control or to pass in a configurable value. Has the following attributes: • type, that defines if the value of the AVP shall be fetched dynamically or statically. Use the keyword Dynamic to fetch the value from the request. Use the keyword Static to defined a static value. • value, that defines what to send in the AVP. If the attribute type is set to Dynamic, use the fully qualified name of the parameter to use. If the attribute type is set to Static, enter the string to use in the AVP. See Defining Static and Dynamic Parameter Mappings.

Table 9-1 Structure and contents of a credit control SLA

Tag	Description
<Amount>	Parent tag: <CCInterception>. Defines how the Diameter AVPs Unit-Value.Exponent and Unit-Value.Value-Digits shall be populated. The alternatives are to either dynamically fetch the value from the request that is being subject to credit control or to pass in a configurable value. • Has the following attributes: • type, that defines if the value of the AVP shall be fetched dynamically or statically. Use the keyword **Dynamic** to fetch the value from the request. Use the keyword **Static** to defined a static value. • value, that defines what to send in the AVP. If the attribute type is set to **Dynamic**, use the fully qualified name of the parameter to use. If the attribute type is set to **Static**, enter the string to use in the AVP. See Defining Static and Dynamic Parameter Mappings.
<Currency>	Parent tag: <CCInterception>. Defines how the Diameter AVP Currency-Code shall be populated. The alternatives are to dynamically fetch the value from the request that is being subject to credit control or to pass in a configurable value. Has the following attributes: • type, that defines if the value of the AVP shall be fetched dynamically or statically. Use the keyword **Dynamic** to fetch the value from the request. Use the keyword **Static** to defined a static value. • value, that defines what to send in the AVP. If the attribute type is set to **Dynamic**, use the fully qualified name of the parameter to use. If the attribute type is set to **Static**, enter the string to use in the AVP. See Defining Static and Dynamic Parameter Mappings.

Table 9-1 Structure and contents of a credit control SLA

Tag	Description
<AsynchronousCommit>	Parent tag: <CCInterception> Defines how and when to trigger charging of a previously reserved amount and which data to use. If present, the charging is asynchronous. If not present, the charging is synchronous. Has the following attributes: • InterfaceName, which defines the Service Gatekeeper interface that the method defined in the attribute methodName belongs to. • methodName, which defines the name of the method that the credit control check applies to. See section Managing and Configuring the Plug-in Manager in *Oracle Communications Services Gatekeeper Administration Guide* for information on how to get a list of interface names and methods. For example, the interface name for Parlay X 2.1 Short Messaging and RESTful Short Messaging call-back interface is com.bea.wlcp.wlng.px21.callback.SmsNotificationCallback and the method that contains a delivery notification is notifySmsReception. Contains: • <ReservationCorrelator>, zero (0) or one (1). • <CommitCorrelator>, exactly one (1). See Correlating Reservation and Commit Triggers in Asynchronous Credit Control Checks.

Table 9-1 Structure and contents of a credit control SLA

Tag	Description
<ReservationCorrelator>	Parent tag: <AsynchronousCommit> Defines what to use as an ID, or correlator, for the reservation part of an asynchronous credit control check. The alternatives are to dynamically fetch the value from the request that is being subject to credit control or to pass in a configurable value. Has the following attributes: • type, that defines if the value of the correlator shall be fetched dynamically or statically. Use the keyword **Dynamic** to fetch the value from the request. Use the keyword **Static** to define a static value. • value, that defines what to use as a correlator. If the attribute type is set to **Dynamic**, use the fully qualified name of the parameter to use. If the attribute type is set to **Static**, enter the string to use. See Defining Static and Dynamic Parameter Mappings. **Note:** The parameter to use is any of the parameters in the method of the reservation request as defined in the tag <CCInterception>. Normally, the correlator used to correlate asynchronous request-response pairs is used.

Table 9-1 Structure and contents of a credit control SLA

Tag	Description
<CommitCorrelator>	Parent tag: <AsynchronousCommit> Defines what to use as an ID, or correlator, for the commit part of an asynchronous credit control check. The alternatives are to dynamically fetch the value from the request that is being subject to credit control or to pass in a configurable value. Has the following attributes: • type, that defines if the value of the correlator shall be fetched dynamically or statically. Use the keyword **Dynamic** to fetch the value from the request. Use the keyword **Static** to define a static value. • value, that defines what to use as a correlator. If the attribute type is set to **Dynamic**, use the fully qualified name of the parameter to use. If the attribute type is set to **Static**, enter the string to use. **Note:** The parameter to use is the parameter of the reservation request as defined in the tag <CCInterception>. Normally, the correlator used to correlate asynchronous request-response pairs is used.

Defining Static and Dynamic Parameter Mappings

The Credit Control Interceptors can map parameters in two ways:

- Static
- Dynamic

Mappings are defined using two components: type and value. The mappings are expressed as attributes to a subset of the Credit Control SLA elements, see Table 9-1.

When type is set to Static, the mapping is static, which means that the attribute value is set to, and treated as an arbitrary string

When type is set to Dynamic, the mapping is dynamic, which means that the attribute value is set to the content of a parameter in the request that is subject to credit control checks. Which parameter to chose is configurable by referring to the Java representation of the parameter defined in the WSDL. See section Managing and Configuring the Plug-in Manager in *Oracle*

Communications Services Gatekeeper Administration Guide for information on how to get a list of valid parameter names. The representation is arg0.<name of parameter as defined in WSDL>.

Depending on which element the parameter mapping exists, the value is passed on to a DIAMTER AVP or used as a key for reservations.

Example of a static parameter mapping:

```
<OCSGChargeDescription type="static" value ="Static value for
reservation."/>
```

Example of a dynamic parameter mapping:

```
<Amount type="dynamic" value ="arg0.charging.amount"/>
```

Correlating Reservation and Commit Triggers in Asynchronous Credit Control Checks

In both synchronous and asynchronous credit control checks, the request that triggers a reservation is defined. This is done using the tag <CCInterception>.

For asynchronous credit control checks, the ID, or correlator, to use to tie together the reservation request and the commit request is defined. This is done in the same manner as parameter mappings are defined, see Defining Static and Dynamic Parameter Mappings. There is one parameter mapping for the reservation, and one for the commit.

In most use cases the correlators are fetched dynamically from the requests. The reservation correlator is fetched from a parameter in the method in the interface defined for the trigger. The commit correlator is fetched from the request that commits the reservation. Since the commit correlator is a parameter in the method that triggers the commit the interface, method, and parameter must be defined explicitly since it is different from the method that triggers the reservations.

For each request, the two correlators are compared and in the case of a match, the reservation is committed. Reservations time-out after a certain time period and the reservation is released. The time-out is defined in the store configuration for the interceptor.

Example Credit Control SLA

Listing 9-1 shows a Credit Control SLA with two different credit control checks.

The firsts check is done asynchronously. The reservation is done when the application sends a sendSms request. In this case, the charging description is a static String, while all other are picked

up dynamically from the request. The reservation is committed when a delivery notification is sent to the application using the method notifySmsReception and the parameter correlator in the sendSms request is identical with the parameter correlator in the notifySmsReception request.

The second check is done synchronously. The reservation is done when the application send a getLocation request. The reservation is committed when the request has successfully returned from the network node.

Listing 9-1 Example Credit Control SLA

```
<?xml version="1.0" encoding="UTF-8"?>
<CCInterceptions>
  <CCInterception
    interfaceName="com.bea.wlcp.wlng.px21.plugin.SendSmsPlugin"
    methodName="sendSms">
    <SubscriptionId type="dynamic" value ="arg0.senderName"/>
    <OCSGChargeDescription
      type="static" value ="Static description for reservation."/>
    <ServiceContextId type="dynamic" value ="arg0.charging.code"/>
    <Amount type="dynamic" value ="arg0.charging.amount"/>
    <Currency type="dynamic" value ="arg0.charging.currency"/>
    <AsynchronousCommit
      interfaceName="com.bea.wlcp.wlng.px21.callback.SmsNotificationCallba
ck"
      methodName="notifySmsReception">
      <ReservationCorrelator
        type="dynamic" value="arg0.receiptRequest.correlator"/>
      <CommitCorrelator type="dynamic" value="arg1.correlator"/>
    </AsynchronousCommit>
  </CCInterception>
```

```
  <CCInterception
    interfaceName="com.bea.wlcp.wlng.px21.plugin.TerminalLocationPlugin"
    methodName="getLocation">
    <SubscriptionId type="dynamic" value="arg0.address"/>
    <OCSGChargeDescription
      type="static" value ="Static dummy description."/>
    <ServiceContextId type="static" value="Static dummy code."/>
    <Amount type="static" value="2"/>
    <Currency type="dynamic" value="USD"/>
  </CCInterception>
</CCInterceptions>
```

Configuration, Management and Provisioning

The Credit Control interceptors are deployed as regular EARs, but they are not registered with the InterceptorManager until a connection is established with the Diameter server. They are de-registered when the connection to the server is connection is closed.

For more information about service interceptors, see chapter Service Interceptors in *Oracle Communications Services Gatekeeper Platform Development Studio - Developer's Guide*.

The interceptors are configured and managed using an MBean.

The Credit Control SLAs are provisioned using the Service Provider Group and Application Group SLA MBeans, see chapter Managing SLAs in *Oracle Communications Services Gatekeeper Managing Accounts and SLAs*.

Properties for Credit Control Service Interceptors

Property	Description
Managed object in Administration Console	<domain name>–>OCSG–><server name>–>ContainerServices–>CreditControlInterceptor
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName=CreditControlInterceptor Type=com.bea.wlcp.wlng.cc_interceptor.management.DiameterMBean
Deployment artifacts	cc_interceptor.ear

Deployment of CreditControlInterceptor

By default the Credit Control interceptors are not deployed.

Deploy the EAR file that contains the interceptors using WebLogic Server tools, see *Oracle WebLogic Server Deploying Applications to WebLogic Server* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/deployment/ for a description of the different deployment options.

The EAR file is located in `$OCSG_HOME/applications`.

Below is an example on how to the deploy the Credit Control Interceptors:

```
java weblogic.Deployer -adminurl http://<admin host>:<admin port> -user
<admin user> -password <admin password -name ccInterpeptor.ear -deploy
```

The interceptors are not connected to the Diameter server after deployment.

Configuration of CreditControlInterceptor

To configure the behavior of the CreditControlInterceptor, in the managed object CreditControlInterceptor:

1. Specify:
   – Attribute: PeerRetryDelay

- Attribute: DestinationHost
- Attribute: DestinationPort
- Attribute: DestinationRealm
- Attribute: OriginHost
- Attribute: OriginPort
- Attribute: MoReservationInterceptorIndex
- Attribute: MtReservationInterceptorIndex
- Attribute: CommitInterceptorIndex
- Attribute: PeerRetryDelay
- Attribute: RequestTimeout
- Attribute: WatchdogTimeout
- Attribute: OriginHost

2. Use Operation: connect to connect to the Diameter server.

Management of CreditControlInterceptor

The Credit Control Interceptors can be explicitly connected to the Diameter server. It does not connect to the server by default. The interceptors have a connection status that will be preserved after redeployment and server restart.

Use:

- Operation: connect
- Operation: disconnect

.Use Attribute: Connected (r) to check the current connection status.

Use Operation: connect after any changes to the configuration attributes. Changes does not take affect until this operation is invoked.

Provision Credit Control SLAs

Credit Control SLAs can be enforced on application group and service provider group level.

The SLAs are provisioned as custom SLAs, see section Managing SLAs in *Managing Accounts and SLAs*.

Before any Credit Control SLAs can be provisioned, the XSD for the SLA must be loaded, and the SLA type is given when the XSD is loaded.

The XSD is found in EAR that contains the Credit Control interceptors, see Deployment of CreditControlInterceptor. The XSD is located in

`/xsd/ccMapping.xsd`

The SLA type to use when loading the Credit Control SLAs XSD is `credit_control`

The SLA type to use when provisioning Credit Control SLAs is `credit_control`

Reference: Attributes and Operations for CreditControlInterceptor

Managed object: Container Services–>CreditControlInterceptor

MBean: com.bea.wlcp.wlng.interceptor.management.DiameterMBean

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

- Attribute: Connected (r)
- Attribute: CommitInterceptorIndex
- Attribute: DestinationHost
- Attribute: DestinationPort
- Attribute: DestinationRealm
- Attribute: OriginHost
- Attribute: OriginPort
- Attribute: PeerRetryDelay
- Attribute: RequestTimeout
- Attribute: WatchdogTimeout
- Operation: connect
- Operation: disconnect

Attribute: Connected (r)

Read-only.

Scope: Server

Format: Boolean

Unit: n/a

Displays the status of the connection to the Diameter server.

Displays:

- `true`, if connected.
- `false`, if not connected.

Attribute: CommitInterceptorIndex

Scope: Cluster

Format: Integer

Unit: n/a

Specifies where in the Service Interceptor chain to register the Credit Control interceptor for committing asynchronous Credit Control Checks.

Attribute: DestinationHost

Scope: Cluster

Format: String

Unit: n/a

Specifies the Destination-Host AVP in the Diameter request.

Can be specified either as an IP-address or a host name.

Attribute: DestinationPort

Scope: Cluster

Format: int

Unit: n/a

Specifies the port on the Diameter server to connect to.

Attribute: DestinationRealm

Scope: Cluster

Format: String

Unit: n/a

Specifies the Destination-Realm AVP in the Diameter request.

Attribute: OriginHost

Scope: Server

Format: String

Unit: n/a

Specifies the Origin-Host AVP in the Diameter request.

Can be specified either as an IP-address or a host name.

Optional.

Attribute: OriginPort

Scope: Server

Format: int

Unit: n/a

Specifies the local originator port to be used for the connection to the Diameter server.

If specified as 0 (zero), a random local port is used.

Random port is a requirement in order to support hitless upgrades of the interceptor.

Attribute: MoReservationInterceptorIndex

Scope: Cluster

Format: int

Unit: n/a

Specifies where in the Service Interceptor chain to register the Credit Control interceptor for network-triggered requests.

Attribute: MtReservationInterceptorIndex

Scope: Cluster

Format: int

Unit: n/a

Specifies where in the Service Interceptor chain to register the Credit Control interceptor for application-initiated requests.

Attribute: PeerRetryDelay

Scope: Cluster

Format: int

Unit: seconds

Specifies the time to wait before attempting to reconnect to the Diameter server if the connection is lost.

Attribute: RequestTimeout

Scope: Cluster

Format: int

Unit: milliseconds

Specifies the maximum time to wait for a response to a request to the Diameter server.

Attribute: WatchdogTimeout

Scope: Cluster

Format: int

Unit: seconds

Specifies the watchdog timeout Tw.

This setting corresponds to the timer Tw, see RFC 3539 at http://www.ietf.org/rfc/rfc3539.txt.

It specifies the time to wait before attempting to reconnect to the Diameter server in the case of a lost connection.

Operation: connect

Scope: Cluster

Connects to the Diameter server.

Once connected, the interceptor will try to reconnect to the Diameter server if the server is restarted or the interceptor is redeployed.

Signature:

`connect()`

Table 9-2 connect

connect	
Parameter	**Description**
-	-

Operation: disconnect

Scope: Cluster

Disconnects from the Diameter server.

Once disconnected, the interceptor will not try to reconnect to the DAMETER server if the server is restarted or the interceptor is redeployed.

Signature:

`disconnect()`

Table 9-3 disconnect

disconnect	
Parameter	**Description**
-	-

CHAPTER 10

Managing and Configuring Statistics and Transaction Licenses

The following sections describe the statistics functionality and the operation and maintenance procedures for statistics for Oracle Communications Services Gatekeeper:

- About Statistics Generation and Reports
- Overview of Statistics Reports
  - System Report to Console
  - System Report to File
  - Weekly System Report
  - Transaction Usage Log Report
- Managing StatisticService
  - Reference: attributes and operations for StatisticsService
- Transaction Types

About Statistics Generation and Reports

Oracle Communications Services 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 calculating usage costs and for grouping reports. Transaction types are in turn grouped into different categories. For more information on transaction types, see *Oracle Communications Services Gatekeeper 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 Oracle Communications Services 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
- Transaction Usage 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 Oracle Communications Services 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.

Transaction Usage Log Report

A transaction usage log (called the license_limit log) can be extracted from Oracle Communications Services Gatekeeper, which records the transactions on which usage costs are based. 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 transaction usage log report is an XML file with a header and a footer.

Listing 10-1 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 Oracle-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. This value is compared with contractual levels by the file auditor to determine usage costs.
- limit: No longer used. Filled with dummy value.
- exceeded: No longer used. Always false.

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 10-2 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 section Transaction Based Licensing in *Oracle Communications Services Gatekeeper 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.
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	Attribute: StoreInterval

Configure Statistics Types and Transaction Types

Statistics types	Use
Add a new	Operation: addStatisticType

Statistics types	Use
Remove an existing	Operation: removeStatisticType
List existing	Operation: listStatisticTypes Operation: listStatisticTypeDescriptors

View In-Flight Statistics counters

To view a snapshot of...	Use
a subset of the statistics counters	Attribute: CounterSnapshot (r)

Generate Statistics Reports

To generate a...	Use
transaction usage log report	Operation: createLicenseLimitLog
weekly report	Operation: createWeeklyReport
statistics summary over a time interval	Operation: getStatistics
statistics summary over the last minutes	Operation: getSystemStatistics
statistics report to file	Operation: saveStatisticsToFile Operation: saveAccountStatisticsToFile

Add Usage Thresholds

To add	Use
a limit alarm threshold for Oracle module based TUPS	Attribute: ModuleBHTUPSThreshold
a limit alarm threshold for platform based TUPS	Attribute: PlatformBHTUPSThreshold

Reference: attributes and operations for StatisticsService

Managed object: Container Services–>StatisticsService

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

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

- Attribute: CounterSnapshot (r)
- Attribute: StoreInterval
- Attribute: ModuleBHTUPSThreshold
- Attribute: PlatformBHTUPSThreshold
- Operation: addStatisticType
- Operation: createLicenseLimitLog
- Operation: createWeeklyReport
- Operation: getStatistics
- Operation: getSystemStatistics
- Operation: listStatisticTypeDescriptors
- Operation: listStatisticTypes
- Operation: removeStatisticType
- Operation: saveAccountStatisticsToFile

- Operation: saveStatisticsToFile

Attribute: CounterSnapshot (r)

Read only.

Scope: Server

Displays a snapshot with information about all statistics-related counters for the chosen Oracle Communications Services 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.

Attribute: ModuleBHTUPSThreshold

Unit: int

Scope: Cluster

Specifies the TUPS limit alarm threshold for Oracle modules. The value of 0 means no alarms.

Attribute: PlatformBHTUPSThreshold

Unit: int

Scope: Cluster

Specifies the TUPS limit alarm threshold for platform modules The value of 0 means no alarms.

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 10-1 addStatisticType

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

Operation: createLicenseLimitLog

Scope: Cluster

Creates a transaction usage 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 10-2 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. **Note:** 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 10-3 createWeeklyReport

createWeeklyReport	
Parameter	**Description**
SpAccountId	Service provider ID to filter on. **Note:** Leave empty to include all service provider account in the report.
StartDate	Specifies the start date for the report. Format is YYYY-MM-DD.
FileName	The filename for the report. Must include an absolute path. The file is created on the local file system of the selected server. **Note:** The file must not already exist. If it 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 10-4 getStatistics

getStatistics	
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 Operation: listStatisticTypes. **Note:** 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. **Note:** 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. **Note:** 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 10-5 getSystemStatistics

getSystemStatistics	
Parameter	**Description**
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 10-6 listStatisticTypeDescriptors

listStatisticTypeDescriptors	
Parameter	**Description**
-	-

Operation: listStatisticTypes

Scope: Cluster

Specifies the statistics types included in the statistics reports.

Signature:

```
listStatisticTypes()
```

Table 10-7 listStatisticTypes

listStatisticTypes	
Parameter	**Description**
-	-

Operation: removeStatisticType

Scope: Cluster

Removes a statistics type.

Signature:

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

Table 10-8 listStatistics

getStatistics	
Parameter	**Description**
`statisticsType`	ID for the statistics type.

Operation: saveAccountStatisticsToFile

Scope: Cluster/Server

Saves a statistics report to file. Filters can be applied on service provider ID and application ID. The report is grouped by account.

Signature:

```
saveAccountStatisticsToFile(StatisticType: String, fromDate: String,
toDate: String, serviceProviderIdentifier String, applicationIdentifier:
String, filename: String)
```

Table 10-9 saveAccountStatisticsToFile

saveStatisticsToFile	
Parameter	**Description**
StatisticsType	ID for the statistics type. **Note:** 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. **Note:** 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. **Note:** Leave empty for all statistics generated from the date and time specified in `fromDate` up to current date and time.
serviceProviderIdentifier	ID of the service provider to filter on. Leave empty for no filtering.
applicationIdentifier	ID of the application to filter on. Leave empty for no filtering.
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. **Note:** The file must not already exist. If it does already exist, the method will fail.

Operation: saveStatisticsToFile

Scope: Cluster/Server

Saves a statistics report to file.

Signature:

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

Table 10-10 saveStatisticsToFile

saveStatisticsToFile	
Parameter	**Description**
serverName	Specifies the name of the server. Leave empty for statistics generated in all servers.
StatisticsType	ID for the statistics type. **Note:** 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. **Note:** 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. **Note:** 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. **Note:** 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 Oracle Communications Services Gatekeeper. The transaction type TRANSACTION_TYPE_EXTENSION is used for new communication services, developed as extensions to Oracle Communications Services Gatekeeper.

Which events generate which statistics for a certain transaction type are described in *Oracle Communications Services Gatekeeper 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 *Oracle Communications Services Gatekeeper Communication Service Reference*.

CHAPTER 11

Setting Up Geographic Redundancy

The following section describes how to set up Oracle Communications Services Gatekeeper 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

The geographic redundancy service replicates data between two geographically distant sites so that applications can switch from one site to another (for example, in case of the catastrophic failure of one) and still have all the configuration data (account information, system SLAs, and budgets) necessary for SLA enforcement available at the second, remote, site. For more information about geographic redundancy, see section Redundancy, Load Balancing, and High Availability chapter in *Oracle Communications Services Gatekeeper Concepts and Architectural Overview*.

The sites are set up in pairs. One member of the pair is designated the geomaster, the other the slave. Each geographic site has a name which is used for looking up data relevant to the site pair. The name of the remote site is defined in the local site.

Configuration Workflow

There are two stages to configuring basic geographic redundancy. Both must be done at each site.

- Configure the geographically redundant service.
- Define the geomaster site

Configure Each Site for Geo-Redundancy

To use geographic redundancy, each site must be appropriately configured. This is accomplished using the GeoRedundantService. Using this service, you:

- Define the ID of the local site in Attribute: GeoSiteId.
- Define the number of failed attempts to reach a remote site before an alarm should be raised in Attribute: RemoteSiteReachabilityAlarmThreshold.
- Define the remote site in Operation: setSiteAddress.

Define the GeoMaster Site

One site of the site pair must be designated the geomaster site. This is accomplished using the GeoStorageService. Using this service, you:

- Define the site that is to serve as the geomaster in Attribute: GeoMasterSiteId

Reference: Attributes and Operations for GeoRedundantService

Managed object: Container Services–>GeoRedundantService

MBean Type:
com.bea.wlcp.wlng.core.budget.management.configuration.GeoRedundantServiceMBean

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

- Attribute: GeoSiteId
- Attribute: RemoteSiteReachabilityAlarmThreshold
- Operation: getSiteAddress
- Operation: listRemoteSites

- Operation: removeSite
- Operation: setSiteAddress

Attribute: GeoSiteId

Scope: Cluster

Format: String

Defines the name of this geographic redundant site. Must be done at both sites. This name is used as key for all operations on the remote site - see Operation: setSiteAddress, Operation: getSiteAddress, Operation: removeSite.

Attribute: RemoteSiteReachabilityAlarmThreshold

Scope: Cluster

Format: int

Specifies the number of attempts made by a site to reach its peer site before raising an alarm. Must be done at both sites.

Whenever the peer sites fail to establish a connection the number of times defined in `RemoteSiteReachabilityAlarmThreshold`, a connection lost alarm is raised.

Operation: getSiteAddress

Scope: Cluster

Signature:

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

Displays the address of a given remote site.

Table 11-1 getSiteAddress

getSiteAddress	
Parameter	**Description**
Site name	The name of the remote site.

Operation: listRemoteSites

Scope: Cluster

Signature:

```
listRemoteSites()
```

Displays a list of registered remote sites.

Table 11-2 listRemoteSites

listRemoteSites	
Parameter	**Description**
-	-

Operation: removeSite

Scope: Cluster

Signature:

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

Removes a site definition for a remote site. If both sites are operational, must be done at both sites

Table 11-3 removeSite

removeSite	
Parameter	**Description**
Site name	The site name of the remote site

Operation: setSiteAddress

Scope: Cluster

Signature:

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

Specifies the address of a remote site. Must be done at both sites

Table 11-4 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

Reference: Attributes and Operations for GeoStorageService

Managed object: Container Services–>GeoStorageService

MBean Type: com.bea.wlcp.wlng.geostorage.management.GeoStorageServiceMBean

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

- Attribute: GeoMasterSiteId
- Operation: syncFromGeoMaster

Attribute: GeoMasterSiteId

Scope: Cluster

Format: String

Defines the geomaster site. This value must be set at both sites and must be one of the two GeoSiteIds set up using the GeoRedundant service. The geomaster keeps the master copy of all geo-configurable data.

Note: If a new site is added to replace a slave site that has failed, it must be added as a slave site. The site that is designated the geomaster site must remain the geomaster site for the lifetime of the site configuration.

If a geomaster site fails permanently, this attribute should be set to empty (temporarily terminating georedundancy) and the failed site should be removed from the configuration using the GeoRedundantService. If a replacement site is added to the configuration, the currently operating site must be the geomaster and the replacement site must be added as the slave.

Operation: syncFromGeoMaster

Scope: Cluster

Signature:

```
syncFromGeoMaster()
```

Forces the slave to resync the account configuration data with the geomaster. Should only be invoked from the slave site. This is used if, for example, the configuration data of the two sites get out of sync, resulting in multiple out-of-sync alarms.

Table 11-5 syncFromGeoMaster

syncFromGeoMaster	
Parameter	**Description**
-	-

Note: This operation potentially copies large amounts of data and therefore should not be used during peak traffic hours.

Managing and Configuring the SNMP service

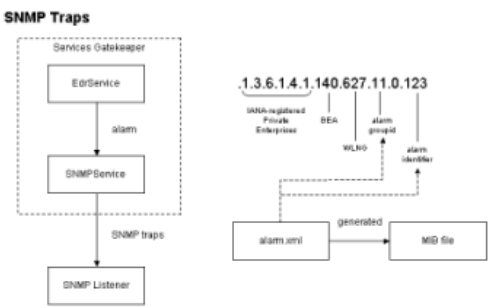
The following section describes how to configure and manage the SNMP service.

- 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 12-1 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 + Oracle Communications Services 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 Oracle Communications Services 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	Attribute: Community
Enterprise Object Identifier. This attribute has a default value and should normally not be changed.	Attribute: EnterpriseObjectIdentifier
SNMP version to use.	Attribute: SNMPVersion
Which alarm severity levels that shall be distributed as SNMP traps.	Attribute: SeverityFilter

Define trap receivers using Operation: addTrapReceiver.

Trap Receivers

To	Use
Add a trap receiver	Operation: addTrapReceiver
List defined trap receivers	Operation: listTrapReceivers
Delete an existing trap receiver	Operation: listTrapReceivers to list a registered trap receivers. Use the ID as an input parameter in Operation: deleteTrapReceiver.

Reference: Attributes and Operations for SNMPService

Managed object: Container Services–>SNMPService

MBean: com.bea.wlcp.wlng.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 Oracle Communications Services 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 12-1 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 12-2 deleteTrapReceiver

deleteTrapReceiver	
Parameter	**Description**
Id	ID of the trap receiver to delete. Use Operation: listTrapReceivers to list the IDs.

Operation: listTrapReceivers

Scope: Cluster

Displays a list of all registered trap receivers.

Signature:

```
listTrapReceivers()
```

Table 12-3 listTrapReceivers

listTrapReceivers	
Parameter	**Description**
-	-

Managing and Configuring the Trace Service

The following section describes how to configure and manage the Trace Service in Oracle Communications Services Gatekeeper:

- 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. Oracle Communications Services Gatekeeper maintains the log file, named default.log. Each service instance writes to this log file, 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 `rootdefault` maintains the trace file.

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 Oracle Communications Services 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 Oracle Communications Services 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.wlng.log.management.TraceServiceMBean

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

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 13-1 addContextCategory

addContextCategory	
Parameter	**Description**
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 com.bea.wlcp.wlng.plugin.sms.smpp

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 13-2 addContextFilter

addContextFilter	
Parameter	**Description**
Identifier	ID of a context trace file.
Type	One of the defined types to filter on. Use: • SERVICE_PROVIDER to filter on service provider ID. • APPLICATION to filter on application ID. • APPLICATION_INSTANCE to filter on application instance ID. • SESSION to filter on session ID. • TRANSACTION to filter on transaction ID.
Value	Value to filter on for this filter type.

Operation: attachAppender

Scope: Cluster

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

Signature:

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

Table 13-3 attachAppender

attachAppender	
Parameter	**Description**
LoggerName	Name of logger.
AppenderName	Name of Oracle Communications Services Gatekeeper internal 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 13-4 createContextTraceFile

createContextTraceFile	
Parameter	**Description**
Identifier	ID of the context trace file to create. Also the name of trace file. The filename has the suffix `.log`. Must be unique per managed server.
Category	The Log4J category.
Threshold	The threshold for the context trace file. Valid values are: • OFF • FATAL • ERROR • WARN • INFO • DEBUG • TRACE • ALL

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 13-5 createRootContextTraceFile

createContextTraceFile	
Parameter	**Description**
Identifier	ID of the context trace file. Also the name of trace file. The filename has the suffix `.log`. 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 13-6 flushBuffers

flushBuffers	
Parameter	**Description**
-	-

Operation: removeContextTraceFile

Scope: Cluster

Removes a context trace file.

Signature:

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

Table 13-7 removeContextTraceFile

removeContextTraceFile	
Parameter	**Description**
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 13-8 resetContextFilters

resetContextFilters	
Parameter	**Description**
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 13-9 rollOver

rollOver	
Parameter	**Description**
-	-

Log4J Hierarchies, Loggers, and Appenders

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

- log4j:hiearchy=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 Oracle Communications Services Gatekeeper Administration Console, in **<Server Name>**–>**Log4J**. For Access Tier servers, there is no management attributes or operations exposed in the Oracle Communications

Services 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 (WSLT) and change to the custom tree where Oracle Communications Services 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=default
- log4j:appender=default,layout=org.apache.log4j.TTCCLayout
- log4j:hiearchy=default
- log4j:logger=rootdefault

For example, to change the trace level for the Access Tier servers for the appender log4j:appender=default to WARN:

1. `cd('log4j:appender=default')`
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 13-1 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"/>
```

```
    <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>
```

CHAPTER 14

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 14-1outlines the store types.

Table 14-1 Store types

Type	Description
Cluster cache	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.
Write behind database store	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. This store has similar performance characteristics as the cluster store for data that is available in the cache, but with better availability.
Write through database store	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. 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.
Database log store	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. 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. 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.

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)

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 14-2 getProviderName

getProviderName	
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 14-3 getStoreSize

getStoreSize	
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 14-4 getTableName

getTableName	
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 14-5 getTypeId

getTypeId	
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 14-6 listStoreNames

listStoreNames	
Parameter	**Description**
-	-

CHAPTER 15

Managing the Policy Service

The following section describes how to manage the Policy Service in Oracle Communications Services Gatekeeper:

- 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 Oracle Communications Services 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 exists 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.wlng.policy.PolicyServiceMBean

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

Operation: deleteApplicationRuleFile

Deletes a rule file on the application group level.

Signature:

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

Table 15-1 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 15-2 deleteServiceProviderRuleFile

deleteServiceProviderRuleFile	
Parameter	**Description**
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 15-3 listApplicationRuleFiles

listApplicationRuleFiles	
Parameter	**Description**
-	-

Operation: listServiceProviderRuleFiles

Displays the IDs of all service provider group rule files.

Signature:

```
listServiceProviderRuleFiles()
```

Table 15-4 listServiceProviderRuleFiles

listServiceProviderRuleFiles	
Parameter	**Description**
-	-

Operation: loadApplicationRules

Loads a rule file on the application group level.

Signature:

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

Table 15-5 loadApplicationRules

loadApplicationRules	
Parameter	**Description**
irlUrl	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 15-6 loadServiceProviderRules

loadServiceProviderRules	
Parameter	**Description**
irlUrl:	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 15-7 viewApplicationRuleFile

viewApplicationRuleFile	
Parameter	**Description**
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 15-8 viewServiceProviderRuleFile

viewServiceProviderRuleFile	
Parameter	**Description**
serviceName	Name of the service for which the rule file is applied.

CHAPTER 16

Setting up WS-Policy and JMX Policy

The following section describes enabling security settings for Web Services and for OAM MBeans in Oracle Communications Services Gatekeeper.

Introduction

One of the first things you must do in setting up your Oracle Communications Services 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 Oracle Communications Services Gatekeeper's interactions with applications. MBean security controls who can have access to the Runtime MBean Server within your WebLogic Server installation, the mechanism that allows OAM procedures to be done.

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, Oracle Communications Services 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.

Oracle Communications Services Gatekeeper uses a WebLogic Server mechanism for Web Services Security -WSSE policies:

- *Oracle WebLogic Server Securing WebLogic Web Services*
  http://download.oracle.com/docs/cd/E12840_01/wls/docs103/webserv_sec/index.html
- *Oracle WebLogic Server Understanding WebLogic Security*
  http://download.oracle.com/docs/cd/E12840_01/wls/docs103/secintro/index.html
- *Web Services Security specifications*
  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 Oracle Communications Services Gatekeeper.

Note: By default, Oracle Communications Services 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 the on-line help for the WebLogic Server administration console for a full description of how to associate a WS-Policy file with a Web Service.

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 `$DOMAIN_HOME/servers/AT1/stage/wlng_at/plan/Plan.xml` (For a single instance development machine, substitute your server name for 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 Oracle Communications Services 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 Oracle Communications Services 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/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 16-1 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 configure identity and trust accordingly. See *Oracle WebLogic Server Securing WebLogic Server* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/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 Oracle Communications Services Gatekeeper.

Starting in the Administrative Console:

1. Configure Default Identity Asserter.

   a. Select **Security realms** –> **my realm** –> **provider** –>**Default IdentityAsserter**

   b. Add **wsse.PasswordDigest** as the active token type.

   c. Modify the data source Name (if not **wlng.datasource**)

2. Configure digest authentication. See the Oracle WebLogic management console on-line help at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/ConsoleHelp/core/index.html for instructions on how to use a password digest in SOAP messages.

3. Create a custom ws-policy xml file for the password digest. See example Listing 16-2 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 16-2 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. Oracle Communications Services 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 Oracle Communications Services Gatekeeper for the first time. Certain configuration values are cached on start up. So, for example, if you started Oracle Communications Services Gatekeeper during the Post-Install phase in order to set up additional JMS Servers, you will need to redeploy the `wlng_at.ear` file after you have made your changes.

To remove the policy files from a Web Service:

1. 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 16-3 and Listing 16-4 for before and after snippets.

Listing 16-3 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 16-4 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 Oracle Communications Services Gatekeeper, you should now re-deploy the EAR file using the Management Console.

Create and use custom a custom WS-Policy

For information about creating and using a custom policy file for message-level security, see *Oracle WebLogic Server Securing WebLogic Web Services* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/webserv_sec.

There is also information about this in the on-line help for Weblogic Server Management Console at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/ConsoleHelp/core/index.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 Oracle Communications Services 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.

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 Oracle Communications Services 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 Oracle Communications Services 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 WebLogic Server Administration Console to do this, as described in the on-line help for the Administration Console at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/ConsoleHelp/core/index.html. 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. Oracle Communications Services Gatekeeper uses the standard WebLogic Server mechanisms for doing this. 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`.

*WebLogic Server Securing WebLogic Resources Using Roles and Policie*s at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/secwlres/index.html contains details on how to use XACML documents to secure WebLogic resources and a reference for XACML on WebLogic Server.

CHAPTER 17

Deployment Model for Communication Services and Container Services

This section describes the deployment model used in Oracle Communications Services Gatekeeper. 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

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, Oracle Communications Services 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 serves as the ser*vice 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 serves as 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/ocsg_4.1/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 HTPP as the transport protocol and handle network-triggered messages from the network node, there is an HTTP servet. HTTP servlets are packaged as Web Archives (WARs) and are packaged in the networ 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 17-1describes the elements in `wlng_nt_<communication service>.ear`.

Listing 17-1 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 EARs.

`META-INF/weblogic-application.xml` is the WebLogic Server-specific deployment descriptor.

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

For more information about enterprise application deployment descriptor elements, see Oracle *WebLogic Server Developing Applications with WebLogic Server* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/programming.

Listing 17-2 describes the elements in a plug-in jar file.

Listing 17-2 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 thing to instantiate the plug-in specific implementation of the plug-in interface using the InstanceFactory. See InstanceFactory in *Platform Development Studio - Developer's Guide f*or more information.

Deployment of SOAP and RESTful Facades on Multiple AT Clusters

For those communication services that have RESTful facades, both SOAP and RESTful facades are included in the standard EAR files. If your installation uses a single cluster of AT servers, this raises no issues. But if your installation needs to use multiple clusters of AT servers, you need to make some adjustments. As a result of certain internal characteristics, it is not possible to deploy the RESTful facade on multiple AT clusters within the same domain. It *is* possible to deploy both the RESTful facade and the SOAP facade on a single cluster within the domain and to deploy only the SOAP facade on multiple other clusters within that same domain. To support this, a special second set of EAR files should be deployed instead of the standard EAR files. These special equivalent EAR versions contain only SOAP interfaces. The file names are identical to those of the standard files except that a `_soap` is appended to the end of the name. The following is a list of the SOAP-only EAR files that should be deployed (as needed) in this situation:

- wlng_at_call_notification_px21_soap.ear

- wlng_at_multimedia_messaging_px21_soap.ear
- wlng_at_payment_px30_soap.ear
- wlng_at_presence_px21_soap.ear
- wlng_at_push_message_ews_soap.ear
- wlng_at_session_soap.ear
- wlng_at_sms_px21_soap.ear
- wlng_at_subscriber_profile_ews_soap.ear
- wlng_at_terminal_location_px21_soap.ear
- wlng_at_third_party_call_px21_soap.ear

These files are found in the `$BEAHOME/ocsg_4.1.1/applications` directory of your installation. Information on the content of these files is available in the "Properties" section of each respective communication services reference in this guide.

This procedure should be done before going into production. It is recommended that you make the console-based changes (all but step 6) with only the Admin Server running. For step 6, you will need to shut down all servers and re-start.

In general these are the steps required. Using the Admin console:

1. Create Managed Servers for the non-REST AT clusters (for example, WLNG_AT1 and WLNG_AT2).
2. Create Managed Servers for the REST_AT cluster (for example, REST_AT1 and REST_AT2)
3. Create the clusters for these servers (for example WLNG_AT_Cluster and REST_AT_Cluster).
4. Assign the Managed Servers to their appropriate clusters:

   WLNG_AT_Cluster WLNG_AT1 WLNG_AT2

   REST_AT_Cluster REST_AT1 REST_AT2
5. Change all deployed applications with REST interfaces (that is, the standard EARs), including session manager, to target REST_AT_Cluster
6. Change the targets of the AT JMS Servers from WLNG_AT* to the REST_AT*

7. Change the target of WLNG_ATJMSResource from WLNG_AT_Cluster to REST_AT_Cluster.
8. Shut down all servers. Carefully edit the config.xml file manually. In default installations this file can be found at `<domain-home>/config/config.xml`.

Listing 17-3 The snippet to edit

```
<custom-resource>
  <name>accesstier</name>
  <target>WLNG_AT_Cluster</target>
  <descriptor-file-name>custom/at.xml</descriptor-file-name>

<resource-class>com.bea.wlcp.wlng.management.descriptor.resource.WlngTierR
esource</resource-class>

<descriptor-bean-class>com.bea.wlcp.wlng.management.descriptor.bean.WlngTi
erBean</descriptor-bean-class>
  </custom-resource>
```

Change the target in the accesstier custom resource from WLNG_AT_Cluster to WLNG_AT_Cluster, REST_AT_Cluster.

9. Start up the Admin Server and install and deploy the SOAP-only EAR files to the non-REST cluster (WLNG_AT_Cluster).
10. Start the newly installed EAR files using the Admin Console. If you forget to do this, their status will be “prepared”.
11. Start your Managed Servers.

For more information on using the Admin Console to accomplish these tasks, click Help on the console screeen. WebLogic Server Administration Console Help comes up.

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 `wlng_at_<communication service>.ear` and `wlng_nt_<communication service>.ear`, respectively.

For more information about how to develop applications for production redeployment, see *Oracle WebLogic Developing Applications with WebLogic Server* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/programming/index.html.

Deploy and Undeploy Communication Services and plug-ins

Individual communication service are deployed and undeployed as EARs.

See *Oracle WebLogic Server Deploying Applications to WebLogic Server* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/deployment/ 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 *Oracle Communications Services 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 are 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 weblogic.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 witch 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/ocsg_4.1/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/`wlsserver_10.3/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 17-1 Arguments to patch build file

Target/Argument	Description
patch	This target applies a patch.
printpatches	This target displays the currently applied patches. The information includes: • Manifest for the EAR • Manifest-Version • Bundle-Name • Created-By • Ant-Version • Weblogic-Application-Version • Bundle-Vendor • Bundle-Version • For each JAR in the EAR, patched class and ID of the patch is displayed.
src	The EAR file to apply the patch to.
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.	`ant printpatches -Dsrc=original.ear`
View patch information for all EARs in specified directory.	`ant printpatches -Dsrc=/path_to_ears`
Apply a single patch to an EAR.	`ant patch -Dsrc=original.ear -Ddest=patched.ear -Dversion=4.0_1 -Dpatchdir=/patches -Dpatchfiles=CR1234.jar`
Apply all patches located in the directory /patches.	`ant patch -Dsrc=original.ear -Ddest=patched.ear -Dversion=4.0_2 -Dpatchdir=/patches -Dpatchfiles=*.jar`

Overview of Container Services and Configuration Files

Table 17-2 gives an overview of the Oracle Communications Services Gatekeeper configuration files.

Table 17-2 Oracle Communications Services Gatekeeper Configuration files

File	Contains configuration for...
$DOMAIN_HOME/config/config.xml	WebLogic Server. See the section describing the domain configuration files in *Oracle WebLogic Server Understanding Domain Configuration at* http://download.oracle.com/docs/cd/E12840_01/wls/docs103/domain_config/ The following additions are specific to Oracle Communications Services Gatekeeper: ...a <custom-resource>, with `<name>accesstier</name>`, that specifies the location of at.xml. ...a <custom-resource>, with `<name>networktier</name>`, that specifies the location of nt.xml. ...A set of <app-deployment> specific for the deployed communication services. See the *Deployment Artifacts* section for the relevant communication service in Communication Service Reference. 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 *Oracle WebLogic Server Deploying Applications to WebLogic Server* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/deployment/
$DOMAIN_HOME/config/custom/at.xml	Oracle Communications Services Gatekeeper Access Tier container and container services. This is a custom resource. **Note:** Do not edit this file.

Table 17-2 Oracle Communications Services Gatekeeper Configuration files

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

Container services

Table 17-3 describes the Oracle Communications Services Gatekeeper container services. The interfaces to these, together with common classes, are packaged in

`$BEA_HOME/ocsg_4.1/server/lib/wlng/wlng.jar`

The container services are packaged in separate JARs, located in `$BEA_HOME/ocsg_4.1/modules`

Table 17-3 Container services

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

Table 17-3 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.georedundant.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.
com.bea.wlcp.wlng.storage.georedundant.GeoRedundantStoreServerService	Storage provider for intercepting store operations for geographically redundant stores, forwarding changes to the geo storage server service functions and reads to the local storage provider.
com.bea.wlcp.wlng.geostorage.GeoStorageServerService	Geo storage server service that has the geo storage singleton services, manages the geo-master lifecycle, performs calls remote calls to the other site, and consistency checks between sites. Also has the geo storage Mbean.
com.bea.wlcp.wlng.tier_routing.TierRoutingManagerServerService	Tier routing manager.

Retired container services

Table 17-4 describes the retired Oracle Communications Services Gatekeeper container services. These services are deprecated and are not deployed as a part of standard Oracle Communications Services Gatekeeper installation. The services are packaged in:

- `wlng_at_bc.ear`, see Table 17-4.
- `wlng_nt_bc.ear`, see Table 17-5.

Table 17-4 Retired services, kept for backwards compatibility in $BEA_HOME/ocsg_4.1/applications/wlng_at_bc.ear

Container service	Summary
Access Web Service	Used by applications to establish sessions. Uses the Access service, see Table 17-5s.

Table 17-5 Retired services, kept for backwards compatibility in $BEA_HOME/ocsg_4.1/applications/wlng_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 Oracle Communications Services 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.wlng.corba.ob.OrbacusServerService</service-class></service>
4. Start server.

Patching of Container Services

Patching of container services is done using SmartUpdate, see *WebLogic Server Installing Patches and Maintenance Packs* at http://download.oracle.com/docs/cd/E12840_01/common/smartupdate/guide/.

Patches to container services and WebLogic Server is done using rolling upgrades, see Oracle *WebLogic Server Upgrading WebLogic Application Environment*s at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/upgrade/index.html

Hitless Upgrade Using Production Redeployment

Oracle Communications Services Gatekeeper supports seamless upgrades of Communication Services and Service Interceptors 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 Oracle Communications Services 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. For more information, see *Oracle WebLogic Server Deploying Applications to WebLogic Server* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/deployment.

Upgrades of container services provided by Oracle Communications Services Gatekeeper can not be done using production redeployment. Rolling upgrades should be used instead. See *Oracle WebLogic Server Upgrading WebLogic Application Environments* at http://download.oracle.com/docs/cd/E12840_01/wls/docs103/upgrade.

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.

Service interceptors supports production redeployment.

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 an Oracle Communications Services Gatekeeper plug-ins and a SIP application. The Oracle Communications Services Gatekeeper side plug-ins can be hitlessly upgraded the same way as other Oracle Communications Services Gatekeeper plug-ins.

  Graceful retirement is not supported for SIP applications. For the SIP Servlet part of the plug-ins, Oracle Communications Services 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 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.

- Native SMPP communication service does not support hitless upgrades. The communication service needs to be undeployed and deployed to be updated.
- Native MM7 communication service handles hitless upgrades as described for HTTP-based network protocol plug-ins.

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 *Oracle Communications Services 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

Oracle Communications Services Gatekeeper uses the WebLogic Server mechanism for production redeployment.

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>
```

CHAPTER 19

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

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)
- 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.
- Routing logic

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

Routing logic is expressed in XML, and is evaluated by means of 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 use data configured using the plug-in manager:

- CreatePluginList
- RemoveInvalidRoute

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.

Plug-in Routing Logic

The plug-in routing logic matches data in a request, and data associated with a request, with routing logic and results in the selection of a plug-in instance. The request is handed off to the selected plug-in instance for further processing.

The routing logic offers a fine grained way to select a network protocol plug-in instance and makes it possible to select a plug-in instance, and thereby a network node, to be targeted for individual requests, based on all data available in the request.

In combination with the plug-in instance feature make it possible to single out a certain network node based on the above operands and thereby:

- offer different QoS levels. Example: different network nodes offers different QoS, for example latencies for message delivery.
- ensure that requests are routed to a node that can actually handle the request. Example: one SMSC is capable of handling binary content in the form of SM logos, while another is not.
- maximize utilization of the network nodes. Example: Two network nodes offers exactly the same functionality, but one processes the request more expensively, so the selection is done based on the charging code the application supplied.

In combination with the plug-in instance feature make it possible to single out a network protocol plug-in instance even if the different plug-in instances connect to the same network node and thereby use different configurations for the request towards the network node.

Example: A network protocol plug-in offers the possibility to configure a priority parameter when sending request to the network node, by setting the parameter differently in different plug-in instances, different priorities can be used for different service providers. In the same way credentials can be mapped so the originator of the request is mapped to the request that is made to the network node.

Defining Routing Logic

The routing logic is expressed in XML and provides:

- a set of logical operators:
  - AND
  - OR
  - NOT
- access to a set of operands:
  - the method name
  - all parameters in the request

- the authentication data, and the data associated with the authentication data, such as service provider ID, application ID, and application instance ID.
- the destination address in the request
- tunnelled parameters provided by the application as xparams in the SOAP header.

The XML based routing logic is specified per plug-in instance and routes requests to a plug-in based on logical expressions and tests that gets evaluated.

The logical operators are XML elements, as described in Table 19-1.

Table 19-1 Logical operators/elements for Plug-in routing

Operator/Element	Description
and	This element contains 1 or many nested elements that in turn are evaluated. This expression will only be true if all the contained elements are true.
or	Contains 1 or many nested elements that in turn are evaluated. This expression will be true if any of the contained elements are true.
not	This element can contain only one nested element. This expression results in the inverse of the nested element.

The operands are XML elements, as described in Table 19-2.

Table 19-2 Operands/elements for Plug-in routing

Operand/Element	Description
spId	The service provider ID associated with the request is compared with the value given in this element. The result is true only if there is an exact match.
appId	The application ID associated with the request is compared with the value given in this element. The result is true only if there is an exact match.

Table 19-2 Operands/elements for Plug-in routing

Operand/Element	Description
appInstanceId	The application instance ID associated with the request is compared with the value given in this element. The result is true only if there is an exact match.
destAddress	The destination address provided in the request is compared with the value given in this element. Which parameter that is considered the destination address is plug-in specific. The format of the value is a regular expression specified as a string. The result is true only if the regular expression matches. See How address ranges are specified when setting up routes for examples of regular expressions. The element has the optional attribute `defaultResult`. It is a Boolean attribute that affects the end result if the request does not contain any destination address. If the attribute is set to false the evaluation results in false. If set to true, the evaluation results in true. Default value is false.

method	The name of the method that the application invoked is compared with the value given in this element.
	The result is true only if there is an exact match.
	Examples:
	`<method>sendSms</method>`
	`<method>makeACall</method>`

Table 19-2 Operands/elements for Plug-in routing

Operand/Element	Description

param	A named parameter in the request is compared with the value given in this element. There are three attributes to this element:
	• `name`, the parameter name. The format of the atttribute is the same as used when referring to parameters in the SLAs. Mandatory attribute.Use Operation: getServiceInfo for a list of parameter names.
	• `value`, the value of the parameter. If the parameter is not of type String, the Java toString() method is used to convert it to a String. Mandatory attribute.
	• `defaultResult`, a Boolean attribute that affects the end result if the request does not contain the specified `name` attribute. If set to false the evaluation results in false if the `name` attribute is not present in the request. If set to true, the evaluation results in true if the if the `name` attribute is not present in the request. Default value is false. Optional attribute.
	The result is true only if there is a match or if `defaultResult` is set to true and the `name` attribute is not present at all in the request.
	The result is true only if there is a match. The value can be expressed as a regular expression.
	Example:
	`<param name="arg0.senderName" value="tel:123456"/>`
	`<param name="arg0.senderName" value="tel:123.*"/>`

Table 19-2 Operands/elements for Plug-in routing

Operand/Element	Description

xparam	A parameter supplied in the request as a tunnelled parameter (xparam) in the request is compared with the value given in this element. There are three attributes to this element:
	• `name`, corresponds to the contents of the `key` attribute in the `param` element in the request. Mandatory attribute.
	• `value`, corresponds to the contents of the `value` attribute in the `param` element in the request.Mandatory attribute.
	• `defaultResult`, a Boolean attribute that affects the end result if the request does not contain the specified `name` attribute. If set to false the evaluation results in false if the `name` attribute is not present in the request. If set to true, the evaluation results in true if the if the `name` attribute is not present in the request. Default value is false. Optional attribute.
	The result is true only if there is a match or if `defaultResult` is set to true and the `name` attribute is not present at all in the request.
	The result is true only if there is a match. The value can be expressed as a regular expression.
	Example:
	The tunneled parameter is defined as:
	`<xparams>`
	`<param key="aParameterName" value="aParameterValue" />`
	`</xparams>`
	A match occurs if the `xparam` element in the routing configuration is
	`<xparam name="aParameterName" value="aParameterValue"/>`

The plug-in routing configuration XML document is validated when it is provisioned. The XML is validated according to the XSD, see Plug-in Routing XSD.

For a set of examples, see Plug-in Routing Configuration Examples.

Plug-in Routing Configuration Examples

The example in Listing 19-1 matches requests sent to an address starting with `tel:123`. If the request does not contain an address this route does not match.

Listing 19-1 Plug-in Routing Configuration Example 1

```
<?xml version="1.0" encoding="UTF-8"?>

<route xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="route.xsd">

  <destAddress>tel:123.*</destAddress>

</route>
```

The example in Listing 19-2 matches any address and also matches any request that does not contain an address.

Listing 19-2 Plug-in Routing Configuration Example 2

```
<?xml version="1.0" encoding="UTF-8"?>

<route xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="route.xsd">

  <destAddress defaultResult="true">.*</destAddress>

</route>
```

The example in Listing 19-3 matches requests sent from the service provider with the ID `sp1`, and the application with the ID `app1`.

Listing 19-3 Plug-in Routing Configuration Example 3

```
<?xml version="1.0" encoding="UTF-8"?>

<route xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="route.xsd">

  <and>

    <spId>sp1</appId>

    <appId>app1</appId>
```

```
  </and>
</route>
```

The example in Listing 19-4 matches all requests except the ones where the operation is `sendSmsRingtone` and either sent from and application using application instance ID `appInst1` or the operation is `sendSms` withe the value of the parameter `senderName` is `tel:123456`.

Listing 19-4 Plug-in Routing Configuration Example 4

```
<?xml version="1.0" encoding="UTF-8"?>
<route xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="route.xsd">
  <and>
    <not><method>sendSmsRingtone</method></not>
    <or>
      <appInstanceId>appInst1</appInstanceId>
      <and>
        <method>sendSms</method>
        <param name="arg0.senderName" value="tel:123456"/>
      </and>
    </or>
  </and>
</route>
```

Plug-in Routing XSD

Below is the XSD to use when creating a plug-in routing configuration XML file.

Listing 19-5 route.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="route">
    <xs:complexType>
      <xs:group ref="ConditionGroup" minOccurs="1" maxOccurs="1"/>
    </xs:complexType>
  </xs:element>

  <xs:group name="ConditionGroup">
    <xs:choice>
      <xs:element name="and" type="AndType"/>
      <xs:element name="or" type="OrType"/>
      <xs:element name="not" type="NotType"/>
      <xs:element name="spId" type="xs:string"/>
      <xs:element name="appId" type="xs:string"/>
      <xs:element name="appInstanceId" type="xs:string"/>
      <xs:element name="destAddress" type="AddressType"/>
      <xs:element name="method" type="xs:string"/>
      <xs:element name="param" type="ParamType"/>
      <xs:element name="xparam" type="ParamType"/>
    </xs:choice>
  </xs:group>

  <xs:complexType name="AndType">
    <xs:group ref="ConditionGroup" minOccurs="1" maxOccurs="unbounded"/>
  </xs:complexType>
```

```
  <xs:complexType name="OrType">
    <xs:group ref="ConditionGroup" minOccurs="1" maxOccurs="unbounded"/>
  </xs:complexType>

  <xs:complexType name="NotType">
    <xs:group ref="ConditionGroup" minOccurs="1" maxOccurs="1"/>
  </xs:complexType>

  <xs:complexType name="AddressType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="defaultResult" type="xs:boolean" default="false"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="ParamType">
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="value" type="xs:string" use="required"/>
    <xs:attribute name="defaultResult" type="xs:boolean" default="false"/>
  </xs:complexType>
</xs:schema>
```

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.

Plug-in Routing Logic and Plug-in Routes

Plug-in routing logic is an extension the plug-in routes and Operation: setRouteConfig and Operation: getRouteConfig replaces Operation: addRoute and Operation: removeRoute.

Plug-in routes added using Operation: addRoute, are converted into plug-in routing logic where route is added as a logical OR given that the route that is being modified only contain `<or>` and `<destAddress defaultResult="true">` elements.

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 Routing Logic and Node IDs

The administration of routes uses the following operations:

- To add new routing logic: Operation: setRouteConfig
- To change or remove routing logic: Operation: getRouteConfig and Operation: setRouteConfig
- To list all 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: getRouteConfig
- 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

Deprecated. Use Operation: setRouteConfig.

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 19-3 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 Operation: listPluginInstances.
AddressExpression	The pattern to be used as a matching criteria. See How address ranges are specified when setting up routes.

Operation: createPluginInstance

Scope: Cluster

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

Signature:

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

Table 19-4 createPluginInstance

createPluginInstance	
Parameter	**Description**
PluginServiceId	ID of the plug-in service to instantiate. See Operation: listPluginServices 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 Operation: listPluginInstances.

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 19-5 destroyPluginInstance

destroyPluginInstance	
Parameter	**Description**
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 19-6 getPluginInstanceInfo

getPluginInstanceInfo	
Parameter	**Description**
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 19-7 getPluginNodeId

getPluginNodeId	
Parameter	**Description**
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 19-8 getPluginServiceInfo

getPluginServiceInfo	
Parameter	**Description**
PluginServiceId	ID of the plug-in service.

Operation: getRouteConfig

Scope: Cluster

Gets the plug-in routing logic XML for a given plug-in instance.

Signature:

```
getRouteConfig(PluginInstanceId :String)
```

Table 19-9 getRouteConfig

getRouteConfig	
Parameter	**Description**
PluginInstanceId	The plug-in instance ID to get the routing configuration for.

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 19-10 getServiceInfo

getServiceInfo	
Parameter	**Description**
Type	ID of the service type. See Operation: listServiceTypes.

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 19-11 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 19-12 listPluginInstances

listPluginInstances	
Parameter	**Description**
-	-

Operation: listRoutes

Scope: Cluster

Displays a list of all registered routes, including the routing logic expressed an XML.

Signature:

```
listRoutes()
```

Table 19-13 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 19-14 listServiceTypes

listServiceTypes	
Parameter	**Description**
-	-

Operation: removeRoute

Scope: Cluster

Deprecated. Use Operation: setRouteConfig.

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 19-15 removeRoute

removeRoute	
Parameter	**Description**
PluginInstanceId	ID of the plug-in.
AddressExpression	The pattern used as a matching criteria. See How address ranges are specified when setting up routes.

Operation: setRouteConfig

Scope: Cluster

Loads plug-in routing logic XML for a plug-in instance. Existing routing logic is overwritten, use Operation: getRouteConfig to get the existing routing logic.

Replaces Operation: addRoute.

Signature:

```
setRouteConfig(PluginInstanceId :String, xmlConfig: String)
```

Table 19-16 setRouteConfig

setRouteConfig	
Parameter	**Description**
PluginInstanceId	The plug-in instance ID to load the routing configuration for.
xmlConfig	The plug-in routing logic. Expressed as XML, see Plug-in Routing Logic.

Operation: setPluginNodeId

Scope: Cluster

Assigns a node ID to a plug-in instance.

Signature:

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

Table 19-17 setPluginNodeId

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

Operation: setRouteConfig

Scope: Cluster

Loads plug-in routing logic XML for a plug-in instance.

Replaces Operation: addRoute.

Signature:

```
setRouteConfig(PluginInstanceId :String, xmlConfig: String)
```

Table 19-18 setRouteConfig

setRouteConfig	
Parameter	**Description**
PluginInstanceId	The plug-in instance ID to load the routing configuration for.
xmlConfig	The plug-in routing logic. Expressed as XML, see Plug-in Routing Logic.

CHAPTER 20

Managing and Configuring the Tier Routing Manager

This chapter describes configuration and maintenance attributes and operations for the Tier Routing Manager. It also provides a workflow for the configuration:

- Introduction
- Configuration Workflow
- Reference: Attributes and Operations for TierRoutingManager

Introduction

Applications can interact with several different types of Service Facades when using Oracle Communications Services Gatekeeper Communication Services. The different types of Service Facades are:

- SOAP
- SOA
- RESTful
- Native

All Service Facades use Service Enablers to connect to the telecom network.

Application-initiated requests are automatically routed from any given Service Facade to the proper Service Enabler, since there is a many-to-one (n:1) relationship between Service Facades and Service Enablers.

Network-triggered requests needs a mechanism to resolve to which Service Facade they shall be routed. The network-triggered case is solved by attaching an identifier to that identifies which Service Facade was used when setting up the subscription to notifications on network triggered requests.

This identifier is used by the Tier Routing Manager to route network-triggered requests to the appropriate Service Facade, or rather to which Access Tier cluster the request shall be passed on to. There is one Access Tier cluster with SOA Service Facades, one cluster with RESTful Service Facades, and one cluster with SOAP Service Facades.

The routing uses *tier routes* to identify to which cluster to route the request. A tier route is identified by an index and has the following properties:

- An endpoint expression
- The name of the cluster for which the route is valid

When an application wishes to receive traffic from the network, it subscribes to notifications for network triggered events. As a part of the subscription, the application provides a URL for the endpoint to which notifications should be sent.

The SOA Service Facades uses the application-provided endpoint URL and rewrites it, embedding information that it was the SOA Service Facade that was used to set up the notification along with the original URL. This new URL is passed used the request to the Service Enabler. The SOA Service Facade replaces the original callback URL with a new URL that points to the Oracle Service Bus Proxy Service and adds the parameter realUrl that has the value of the original callback URL.

For example, if the original callback URL is:

http://somehost/services/SmsNotification

the rewritten callback URL is

http://soahost/proxySms/SmsNotification?realUrl=http://somehost/services/SmsNotification

The RESTful Service Facades always gets a callback URL starting with /bayeux/ from the application, so the URL itself provides information that the subscription originated from the RESTful Service Facade.

When a network-triggered request arrives at the Tier Routing Manager, it inspects the endpoint (rewritten) URL to extract from it the Service Facade to which it should be routed. It looks at the pattern of the URL and matches it to the configured endpoint expression type. When a match is found, it resolves the cluster name and passes the request on to the appropriate cluster.

Note: Some network protocol plug-ins support off-line provisioning for subscriptions for notifications. The callback URL used for these subscriptions must be formatted according to the URL rewrite pattern of the appropriate Service Facade.

See Table 20-1 for examples of endpoint expressions.

Table 20-1 Examples of tier routes

Endpoint expressions	Description
.*realUrl=.*	Routes to a SOA Service Facade cluster, since the parameter realURL is present in the callback endpoint.
/bayeux.*	Routes to a RESTful Service Facade cluster, since the parameter bayeux is present in the callback endpoint.
http://.*mydomain.com/*	Routes to a specific cluster when the callback endpoint is within the domain mydomain.com. This makes it possible to inspect in which domain the callback endpoint is, and use one access cluster for a given set of service providers and another for another set of service providers. It makes it possible to use one Access tier cluster for applications residing in the network operator's intranet and another for applications hosted by service providers outside the network operator's domain.

Configuration Workflow

The administration of routes uses the following operations:

- To add a new tier route: Operation: addTierRoute
- To remove a tier route: Operation: removeTierRoute
- To list all existing tier routes: Operation: listTierRoutes

Reference: Attributes and Operations for TierRoutingManager

Managed object: Container Services–>TierRoutingManagerMBean

MBean: com.bea.wlcp.wlng.tier_routing.TierRoutingManagerMBean

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

- Operation: addTierRoute

Operation: addTierRoute

Scope: Cluster

Adds a new tier route. A route is identified by an index.

Signature:

```
addTierRoute(endpointExpression : String, clusterName: String, index: int)
```

Table 20-2 addTierRoute

addTierRoute	
Parameter	**Description**
endpointExpression	The pattern to be used as a matching criteria for the callback URL for network-triggered operations. Each Service Facade adds its own signature to the pattern. See Table 20-1 for examples of routes. The pattern is a regular expression.
clusterName	The name of the cluster to pass the request to if the endpointExpression matches.
index	The index of the tier route. The first matching route will be used so the order of the routes are important if a URL can match multiple routes. The new entry will shift any existing entries one index higher. To insert entry first in the list, use index 0 (zero). To insert entry last in the list, use the number of list entries (the size of the list).

Operation: listTierRoutes

Scope: Cluster

Displays the list of tier routes. The list includes:

- index

- endpoint expression
- cluster name

Signature:

```
listTierRoutes()
```

Table 20-3 listTierRoutes

listTierRoutes	
Parameter	**Description**
-	-

Operation: removeTierRoute

Scope: Cluster

Removes a tier route. The route is identified by the index. After the update, the index of all tier routes with a higher index than the removed are decreased by one.

Signature:

```
removeTierRoute(index: int)
```

Table 20-4 removeRoute

removeRoute	
Parameter	**Description**
index	Index if the tier route to remove.

CHAPTER 21

Managing and Configuring Third Party Call Communication Services

The following section describes configuration and maintenance attributes and operations for the Oracle Communications Services Gatekeeper 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	Properties for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control
Configuration workflow	Configuration workflow for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control
List of operations and attributes related to management and provisioning	Management Operations for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control
Reference of management attributes and operations	Reference: Attributes and Operations for Parlay X 3.0 Third Party Call/Parlay MultiParty Call Control

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>–>OCSG–><server name>–>Communication Services–>Plugin_px30_third_party_call_parlay_mpcc
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName=Plugin_px30_third_party_call_parlay_mpcc Type=com.bea.wlcp.wlng.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.wlng.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 wlng_nt_third_party_call_px30.ear px30_third_party_call.war, packaged in wlng_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)

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.

Us 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 ifP_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 alllowed 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 alllowed 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 21-1 configCallGetInfoReq

configCallGetInfoReq	
Parameter	**Description**
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: • **True** to add it to TpCallInfoType • **False** to not
PCallInfoReleaseCause	Specifies if the P_CALL_INFO_RELEASE_CAUSE tag shall be present in TpCallInfoType or not. Use: • **True** to add it to TpCallInfoType • **False** to not

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 21-2 configLegGetInfoReq

configLegGetInfoReq	
Parameter	**Description**
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: • **True** to add it to TpCallLegInfoType • **False** to not
PCallLegInfoReleaseCause	Specifies if the P_CALL_LEG_INFO_RELEASE_CAUSE tag shall be present in TpCallLegInfoType or not. Use: • **True** to add it to TpCallLegInfoType • **False** to not
PCallLegInfoAddress	Specifies if the P_CALL_LEG_INFO_ADDRESS tag shall be present in TpCallLegInfoType or not. Use: • **True** to add it to TpCallLegInfoType • **False** to not
PCallLegInfoAppInfo	Specifies if the P_CALL_LEG_INFO_APPINFO tag shall be present in TpCallLegInfoType or not. Use: • **True** to add it to TpCallLegInfoType • **False** to not

Operation: getCallLegs

Scope: Cluster

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

Signature:

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

Table 21-3 getCallLegs

getCallLegs	
Parameter	**Description**
CallSessionId	ID if 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 informnation 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 21-4 getCallSessionInfo

getCallSessionInfo	
Parameter	**Description**
CallSessionId	ID if 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 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.

- `callPariticipantTerminationCause`: 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 21-5 getCallLegSessionInfo

getCallLegSessionInfo	
Parameter	**Description**
CallLegSessionId	ID if 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 21-6 listCallSessionIds

listCallSessionIds	
Parameter	**Description**
-	-

Operation: countPendingCallSession

Scope: Server

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

Signature:

```
countPendingCallSession()
```

Table 21-7 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	Properties for Parlay X 2.1 Third Party Call/INAP
Configuration workflow	Configuration Workflow for Parlay X 2.1 Third Party Call/INAP
List of operations and attributes related to management and provisioning	Management for Parlay X 2.1 Third Party Call/INAP
Reference of management attributes and operations	Reference: Attributes and Operations for Parlay X 2.1 Third Party Call/INAP

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>–>OCSG–><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

Property	Description
Service type	ThirdPartyCall
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 NT EAR wlng_nt_third_party_call_px21.ear	Plugin_px21_third_party_call_inap.jar. Plugin_px21_third_party_call_sip.jar, px21_third_party_call_service.jar, and px21_third_party_call_sip.war
AT EAR: Normal wlng_at_third_party_call_px21.ear	px21_third_party_call.war and rest_third_party_call.war
AT EAR: SOAP Only wlng_at_third_party_call_px21_soap.ear	px21_third_party_call.war

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: 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.

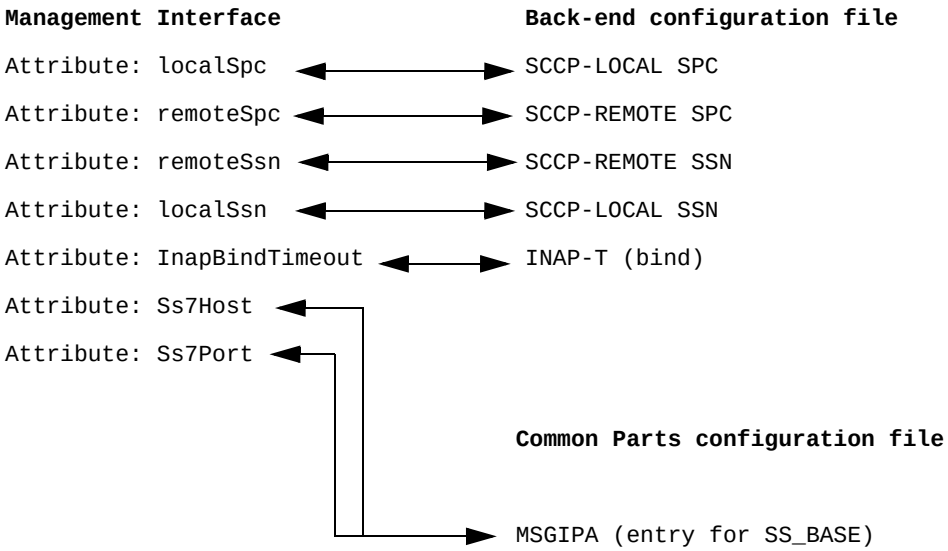
Configuration files and dependencies

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 21-1below presents an overview of the dependencies.

Figure 21-1 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 21-9. 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 Oracle Communications Services 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 21-8 INAP API properties used

Property	Comments
priority	SCCP Message priority.
quality-of-service	SCCP QoS.
inap-user-id	The common parts module ID used by the plug-in instance. See Common parts configuration file. Either INAPUP or any of the USERxx IDs are to be used by the plug-in instance. The numeric identifier of the ID shall be used, not the ID itself (as used in the common parts configuration file). The module ID numeric values can be found in `/opt/ss7/ss7_ITU/include/portss7.h` in an installed stack. USER01 has decimal value 40, USER02 41, and so on.
ss7host	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 Common parts configuration file. 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.
port-number	The port number to which the SS7_BASE module ID is bound in the common parts configuration file.

Table 21-8 INAP API properties used

Property	Comments
bind-timeout	Time to wait for bind response before a bind operation is considered a failure. Unit: milliseconds.
heartbeat-interval	Heartbeat interval between the INAP API and the common parts module. Must correspond to the MSGHBRATE and MSGHBLOST properties defined in the common parts configuration file. See Common parts configuration file. **Note:** 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. Unit: milliseconds.

Listing 21-1 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 21-9. 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 21-9 Common parts configuration file properties with dependencies to plug-in instance settings

Property	Comments
MSGIPA	There must be one MSGIPA entry per plug-in instance. First, choose a Message Port owner ID (MP OwnerID). Use one of the following: • INAPUP • USER01 • USER02 • USER03 • USER04 • USER05 • USER06 • USER07 • USER08 • USER09 • USER10 MP OwnerID shall correspond to inap-user-id specified in Attribute: InapUserId The IP-address (or host name) with TCP port number must correspond to the host where the plug-in instance is deployed. Make sure there is a MSGINTERACT entry per MP OwnerID. Instances of MP OwnerIDs are not supported. Example: MSGIPA=USER01,192.168.20.2:6701 10.10.10.11:6701
MSGHBLOST	Must correspond to Attribute: LocalSpc.
MSGHBRATE	Must correspond to heartbeat-interval in INAP API configuration file

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 21-9. 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 21-10 Back-end configuration file properties with dependencies to plug-in instance settings

Property	Comments
INAP-T (bind)	Must correspond to Attribute: InapBindTimeout given in Plugin_third_party_call_inap.
SCCP-LOCAL SPC	Must correspond to Attribute: LocalSpc given in Plugin_third_party_call_inap.
SCCP-LOCAL SSN	Must correspond to Attribute: LocalSsn given in Plugin_third_party_call_inap.
SCCP-REMOTE SPC	Must correspond to Attribute: RemoteSpc given in Plugin_third_party_call_inap.
SCCP-REMOTE SSN	Must correspond to Attribute: RemoteSsn given in Plugin_third_party_call_inap.

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 a network protocol plug-in instance in Oracle Communications Services 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	Properties for Parlay X 2.1 Third Party Call/SIP
Configuration workflow	Configuration Workflow for Parlay X 2.1 Third Party Call/SIP
List of operations and attributes related to management and provisioning	Management for Parlay X 2.1 Third Party Call/SIP
Reference of management attributes and operations	Reference: Attributes and Operations for Parlay X 2.1 Third Party Call/SIP

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: ControllerURI
   - Attribute: ISCRouteURI
   - Attribute: MaximumCallLength
   - Attribute: StatusRetentionTime
3. 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.
4. 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>–>OCSG–><server name>–>Communication Services–>Plugin_third_party_call_sip
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName=Plugin_px21_third_party_call_sip Type=com.bea.wlcp.wlng.plugin.tpc.sip.management.TPCMBean
Network protocol plug-in service ID	Plugin_px21_third_party_call_sip
Network protocol plug-in instance ID	Plugin_px21_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
Interfaces with the network nodes using:	SIP: Session Initiation Protocol, RFC 3261
Deployment artifacts	px21_third_party_call_service.jar, Plugin_px21_third_party_call_sip.jar and Plugin_px21_third_party_call_sip.war, 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.

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

Scope: Cluster

Unit: n/a

Format: boolean

Specifies if charging is allowed or not. That is, if the parameter charging is allowed to be present in a request from an application.

Attribute: ControllerURI

Scope: Cluster

Unit: n/a

Format: String in URI format

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

Attribute: ISCRouteURI

Scope: Cluster

Unit: n/a

Format: String in URI format

Specifies the URI of the IMS service control route.

Attribute: MaximumCallLength

Scope: Cluster

Unit: minutes

Format: int

Specifies for how long time a call is allowed to be ongoing. If this time expires, the call is terminated.

Attribute: StatusRetentionTime

Scope: Cluster

Unit: minutes

Format: int

Specifies for how long time to retain status information about the call after it has been terminated.

CHAPTER 22

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

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	Properties for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control
Configuration workflow	Configuration Workflow for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control
List of operations and attributes related to management and provisioning	Management and Provisioning Operations for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control
Reference of management attributes and operations	Reference: Attributes and Operations for Parlay X 3.0 Call Notification/Parlay MultiParty Call Control

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>–>OCSG–><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

Property	Description
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.wlng.px30.plugin.callnotification.parlay.management.mbean.CallNotificationMBean

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

Operation: deleteMediaNotification

Scope: Cluster

Deletes a media notification.

Signature:

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

Table 22-1 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 22-2 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.

Signature:

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

Table 22-3 getMediaNotification

getMediaNotification	
Parameter	**Description**
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 `notificationsIds` 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 22-4 getNotification

getNotification	
Parameter	**Description**
correlator	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 22-5 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

Returns a list of correlators that uniquely identify each notification.

Signature:

```
listMediaNotifications()
```

Table 22-6 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 Oracle Communications Services 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	Properties for Parlay X 2.1 Call Notification/SIP
Configuration workflow	Properties for Parlay X 2.1 Call Notification/SIP
List of operations and attributes related to management and provisioning	Management for Parlay X 2.1 Call Notification/SIP
Reference of management attributes and operations	Reference: Attributes and Operations Parlay X 2.1 Call Notification/SIP

Properties for Parlay X 2.1 Call Notification/SIP

Property	Description
Managed object in Administration Console	<domain name>–>OCSG–><server name>–>Communication Services–>Plugin_px21_call_notification_sip
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName=Plugin_px21_call_notification_sip Type=com.bea.wlcp.wlng.plugin.callnotification.sip.management.CallNotificationMBean
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
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 NT EAR wlng_nt_call_notification_px21.ea	px21_call_notification_service.jar, Plugin_px21_call_notification_sip.jar, and px21_call_notification_sip.war

Property	Description
AT EAR: Normal wlng_at_call_notification_px21.ear	px21_call_notification.war, px21_call_notification_callback.jar, and rest_call_notification.war
AT EAR: SOAP Only wlng_at_call_notification_px21_soap.ear	px21_call_notification.war and px21_call_notification_callback.jar

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 Oracle Communications Services 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. 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 22-7 getCallDirectionSubscription

getCallDirectionSubscription	
Parameter	**Description**
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 22-8 getCallDirectionSubscription

getCallDirectionSubscription	
Parameter	**Description**
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 22-9 listCallDirectionSubscriptions

listCallDirectionSubscriptions	
Parameter	**Description**
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 22-10 listNotificationSubscriptions

listNotificationSubscriptions	
Parameter	**Description**
Offset	Start of offset.
Length	Number of entries returned.

Operation: removeAllCallDirectionSubscriptions

Scope: Cluster

Removes all call direction subscriptions.

Signature:

```
removeAllCallDirectionSubscriptions()
```

Table 22-11 removeAllCallDirectionSubscriptions

removeAllCallDirectionSubscriptions	
Parameter	**Description**
-	-

Operation: removeAllNotificationSubscriptions

Scope: Cluster

Removes all call notification subscriptions.

Signature:

```
removeAllNotificationSubscriptions()
```

Table 22-12 removeAllNotificationSubscriptions

removeAllNotificationSubscriptions	
Parameter	**Description**
-	-

Operation: removeCallDirectionSubscription

Scope: Cluster

Removes a call direction subscription.

Signature:

```
removeCallDirectionSubscription(Correlator: String)
```

Table 22-13 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 22-14 removeNotificationSubscription

removeNotificationSubscription	
Parameter	**Description**
Correlator	ID for the subscription. Given by an application when the subscription was started.

CHAPTER 23

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

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	Properties for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP
Configuration workflow	Configuration Workflow for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP
List of operations and attributes related to management and provisioning	Management Operations for Parlay X 2.1 Short Messaging/SMPP and Extended Web Services Binary SMS/SMPP
Reference of management attributes and operations	Reference: Attributes and Operations for 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

Property	Description
Managed object in Administration Console	<domain name>–>OCSG–><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 Managing and Configuring the Plug-in Manager.
Supported Address Scheme	tel
North interfaces	Parlay X 2.1 Short Messaging/SMPP: • com.bea.wlcp.wlng.px21.plugin.SendSmsPlugin • com.bea.wlcp.wlng.px21.plugin.ReceiveSmsPlugin • com.bea.wlcp.wlng.px21.plugin.SmsNotificationManagerPlugin • com.bea.wlcp.wlng.px21.callback.SmsNotificationCallback Extended Web Services Binary SMS/SMPP: • com.bea.wlcp.wlng.ews.plugin.BinarySmsPlugin • com.bea.wlcp.wlng.ews.plugin.BinarySmsNotificationManagerPlugin • com.bea.wlcp.wlng.ews.callback.BinarySmsNotificationCallback
Service type	Sms
Exposes to the service communication layer a Java representation of:	Parlay X 2.1 Short Messaging/SMPP: • Parlay X 2.1 Part 4: Short Messaging Extended Web Services Binary SMS/SMPP: • Extended Web Services Binary SMS
Interfaces with the network nodes using:	SMPP 3.4
Deployment artifacts NT EAR wlng_nt_sms_px21.ear	Parlay X 2.1 Short Messaging/SMPP: Plugin_px21_short_messaging_smpp.jar and px21_sms_service.jar Extended Web Services Binary SMS/SMPP: NT EAR ews_binary_sms_service.jar

Property	Description
AT EAR: Normal wlng_at_sms_px21.ear	Parlay X 2.1 Short Messaging/SMPP px21_sms.war, px21_sms_callback.jar, and rest_sms.war Extended Web Services Binary SMS/SMPP: ews_binary_sms.war and ews_binary_sms_callback.jar
AT EAR: SOAP only wlng_at_sms_px21_soap.ear	Parlay X 2.1 Short Messaging/SMPP px21_sms.war and px21_sms_callback.jar Extended Web Services Binary SMS/SMPP: ews_binary_sms.war and ews_binary_sms_callback.jar

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 Oracle Communications Services 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:
   - 8 for USC2
   - 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:
   - Operation: addOriginatingAddressTypeMapping
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: WindowSize
- Attribute: ModuleId (r)
- Attribute: EsmePassword
- Attribute: EsmeAddressRange
- Attribute: SequenceNumberRangeStartId
- Attribute: SequenceNumberRangeEndId
- Attribute: BindType
- Attribute: ReceiveSmsIgnoreMissingSegments
- Attribute: DataSm
- Attribute: SmscAddress
- Attribute: DefaultDataCoding
- Attribute: SMSCDefaultAlphabet
- Attribute: UserTextMaxLength
- Attribute: SegmentsLimit
- Attribute: MobileCountryCode
- Attribute: MobileNetworkCode
- Attribute: ReceiveSegmentsWaitTime
- Attribute: Messaging Mode
- Attribute: SmscPort
- Attribute: TransmitterProxyLocalAddressAndPort
- Attribute: ReceiverProxyLocalAddressAndPort
- Attribute: TransceiverProxyLocalAddressAndPort
- Attribute: EsmeSystemId
- Attribute: EsmeSystemType
- Attribute: EsmeTon

- Attribute: OriginatingAddressTon
- Attribute: DestinationAddressTon
- Attribute: ForwardXParams
- Attribute: EsmeNpi
- Attribute: OriginatingAddressNpi
- Attribute: DestinationAddressNpi
- Attribute: SmppVersion (r)
- Attribute: EnquireLinkTimerValue
- Attribute: ConnectTotalTimeValue
- Attribute: ConnectDelayValue
- Attribute: RequestTimerValue
- Attribute: EnquireLinkRequestTimerValue
- Attribute: RequestDeliveryReports
- Attribute: ActiveStatus (r)
- Attribute: NumberTransmitterConnections
- Attribute: NumberReceiverConnections
- Attribute: NumberTransceiverConnections
- 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: 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: ModuleId (r)

Scope: Server

Unit: n/a

Format: string

Specifies the plugin instance ID. Read only.

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: 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: SequenceNumberRangeStartId

Scope: Server

Unit: n/a

Format: int

Specifies the start ID of the Sequence Number range.

Attribute: SequenceNumberRangeEndId

Scope: Server

Unit: n/a

Format: int

Specifies the end ID of the Sequence Number range.

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: 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: DataSm

Scope: Server

Unit: n/a

Format: Boolean [TRUE | FALSE]

Specifies if binary data is being sent

- **true** if binary data is being sent
- **false** otherwise

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: 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

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: UserTextMaxLength

Scope: Server

Unit: n/a

Format: int

Specifies the maximum number of characters allowed in a Short Message.

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: 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: 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: Messaging Mode

Scope: Server

Units: n/a

Format: int

Specifies the The ESM_CLASS Messaging Mode for packets.

- **2**: Forward (Transaction mode). Used only when **DataSm** is true.
- **3**: Store and Forward mode

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: 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

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: 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: 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 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: OriginatingAddressTon

Scope: Server

Unit: n/a

Format: Integer

Specifies the ESME Type Of Number (TON). Used as a default for originating address.

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: DestinationAddressTon

Scope: Server

Unit: n/a

Format: Integer

Specifies the ESME Type Of Number (TON). Used as a default for destination address.

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 applications for network triggered requests or not. Use:

- **true** to enable forwarding
- **false** to disable forwarding

Attribute: EsmeNpi

Scope: Server

Unit: n/a

Format: int

Specifies the ESME Numbering Plan Indicator (NPI).

Used 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: OriginatingAddressNpi

Scope: Server

Unit: n/a

Format: int

Specifies the ESME Numbering Plan Indicator (NPI).

Used as a default for originating address.

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: DestinationAddressNpi

Scope: Server

Unit: n/a

Format: int

Specifies the ESME Numbering Plan Indicator (NPI).

Used as a default for destination address.

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: SmppVersion (r)

Read only.

Displays the version of the SMPP protocol being used.

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: 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.

- **8** for USC2

Attribute: ConnectDelayValue

Scope: Server

Unit: seconds

Format: int

Specifies the delay time between connection attempts, when the connection to the SMSC is lost.

Attribute: RequestTimerValue

Scope: Server

Unit: milliseconds

Format: int

Specifies the value of the timer used when sending requests to the SMSC.

Attribute: EnquireLinkRequestTimerValue

Scope: Server

Unit: milliseconds

Format: int

Specifies the value of the timer used when sending Enquire Link requests.

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: 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: 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: 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: 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.wlng.plugin.sms.OriginatingAddressType`, is a 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 23-1 addOriginatingAddressTypeMapping

addOriginatingAddressTypeMapping	
Parameter	**Description**
type	Specifies the originating address type to be mapped.

Table 23-1 addOriginatingAddressTypeMapping

addOriginatingAddressTypeMapping	
Parameter	**Description**
ton	Specifies the ESME Type Of Number (TON). Use: • **0** for UNKNOWN • **1** for INTERNATIONAL • **2** for NATIONAL • **3** for NETWORK • **4** for SUBSCRIBER • **5** for ALPHANUMERIC • **6** for ABBREVIATED
npi	Specifies the ESME Numbering Plan Indicator (NPI). 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

Operation: removeOriginatingAddressTypeMapping

Scope: Cluster

Removes an existing TON/NPI address type mapping for a given originating address type.

Signature:

```
removeOriginatingAddressTypeMapping(type: String)
```

Table 23-2 removeOnlineNotificationInfo

removeOriginatingAddressTypeMapping	
Parameter	**Description**
type	Originating address type for the mapping, see Operation: addOriginatingAddressTypeMapping and Operation: listOriginatingAddressTypeMappings.

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 23-3 countOfflineNotificationCache

countOfflineNotificationCache	
Parameter	**Description**
-	-

Operation: countOnlineNotificationCache

Read only.

Scope: Cluster

Displays the number of entries in the on-line notification cache.

Signature:

```
countOnlineNotificationCache()
```

Table 23-4 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 23-5 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 Oracle Communications Services 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 23-6 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 23-7 getOfflineNotificationInfo

getOfflineNotificationInfo	
Parameter	**Description**
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 23-8 getOnlineNotificationInfo

getOnlineNotificationInfo	
Parameter	**Description**
correlator	Correlator identifying the notification.

Operation: listOfflineNotificationInfo

Read only.

Scope: Cluster

Displays a list of all notifications registered off-line.

Signature:

```
listOfflineNotificationInfo()
```

Table 23-9 listOfflineNotificationInfo

listOfflineNotificationInfo	
Parameter	**Description**
-	-

Operation: listOnlineNotificationInfo

Read only.

Scope: Cluster

Displays a list of all notifications registered by an application.

Signature:

```
listOnlineNotificationInfo()
```

Table 23-10 listOnlineNotificationInfo

listOnlineNotificationInfo	
Parameter	**Description**
-	-

Operation: listOriginatingAddressTypeMappings

Read only.

Scope: Cluster

Displays a list of all originating address type mappings.

Signature:

```
listOriginatingAddressTypeMappings()
```

Table 23-11 listOriginatingAddressTypeMappings

listOriginatingAddressTypeMappings	
Parameter	**Description**
-	-

Operation: removeOfflineNotificationInfo

Scope: Cluster

Removes a notification registered off-line.

Signature:

```
removeOfflineNotificationInfo(correlator: String)
```

Table 23-12 removeOfflineNotificationInfo

removeOfflineNotificationInfo	
Parameter	**Description**
correlator	Correlator identifying the notification.

Operation: removeOnlineNotificationInfo

Scope: Cluster

Removes a notification registered by an application.

Signature:

```
removeOnlineNotificationInfo(correlator: String)
```

Table 23-13 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 23-14 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 23-15 startSmsNotification

startSmsNotification	
Parameter	**Description**
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.

CHAPTER 24

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

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	Properties for Parlay X 2.1 MultiMedia Messaging/MM7
Configuration workflow	Configuration Workflow for Parlay X 2.1 MultiMedia Messaging/MM7

To see a	Refer to
List of operations and attributes related to management and provisioning	Provisioning Workflow for Parlay X 2.1 MultiMedia Messaging/MM7
Reference of management attributes and operations	Reference: Attributes and Operations for Parlay X 2.1 MultiMedia Messaging/MM7

Properties for Parlay X 2.1 MultiMedia Messaging/MM7

Property	Description
Managed object in Administration Console	<domain name>–>OCSG–><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 Managing and Configuring the Plug-in Manager.
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 NT EAR wlng_nt_multimedia_messaging_px21.ear	Plugin_px21_multimedia_messaging_mm7.jar, px21_multimedia_messaging_service.jar, multimedia_messaging_mm7_rel5mm712.war, and multimedia_messaging_mm7_rel5mm715.war
AT EAR: Normal wlng_at_multimedia_messaging_px21.ear	px21_multimedia_messaging_callback.jar, px21_multimedia_messaging.war, and rest_multimedia_messaging.war
AT EAR: SOAP Only wlng_at_multimedia_messaging_px21_soap.ear	px21_multimedia_messaging_callback.jar and px21_multimedia_messaging.war

Configuration Workflow for Parlay X 2.1 MultiMedia Messaging/MM7

Below is an outline for configuring the plug-in using the Oracle Communications Services 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://<IP Address of NT server>:<port>/<server>:<port>/<context-root>/<Plug-in instance ID>`

   If using the REL-5-MM7-1-2 XSD, default `<context-root>` is `mmm-mm7`

   If using the REL-5-MM7-1-5 XSD, `<context-root>` is `mmm-mm7-rel5mm7-1-5`
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 Oracle Communications Services 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.wlng.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, REL-5-MM7-1-5]

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.
- REL-5-MM7-1-5, to use REL-5-MM7-1-5.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, critieria: String, appInstanceID:
String)
```

Table 24-1 enableReceiveMms

enableReceiveMms	
Parameter	**Description**
shortcode	The destination address or service activation number of the Multmedia message. Prefixed with the URI, for example `tel:`
critieria	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 24-2 getOfflineNotificationInfo

getOfflineNotificationInfo	
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 24-3 getOfflineNotificationInfo

getOfflineNotificationInfo	
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 24-4 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 24-5 listOnlineNotificationInfo

listOnlineNotificationInfo	
Parameter	**Description**
-	-

Operation: removeOfflineNotificationInfo

Scope: Cluster

Removes a notification registered offline using Operation: enableReceiveMms.

Signature:

```
removeOfflineNotificationInfo(Registration Identifier: String)
```

Table 24-6 removeOfflineNotificationInfo

removeOfflineNotificationInfo	
Parameter	**Description**
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 24-7 removeOnlineNotificationInfo

removeOnlineNotificationInfo	
Parameter	**Description**
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, critieria:
String, appInstanceID: String)
```

Table 24-8 startMessageNotification

startMessageNotification	
Parameter	**Description**
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.

CHAPTER 25

Managing and Configuring Payment Communication Services

The following section describes configuration and maintenance attributes and operations for the communication services that expose Parlay X 3.0 Payment Web Services interfaces. The sections also provide a workflow for the configuration:

- Parlay X 3.0 Payment /Diameter
  - Properties for Parlay X 3.0 Payment/Diameter
  - Configuration Workflow for Parlay X 3.0 Payment/Diameter
  - Provisioning Workflow for Parlay X 3.0 Payment/Diameter
  - Reference: Attributes and Operations for Parlay X 3.0 Payment/Diameter

Parlay X 3.0 Payment /Diameter

This section contains a description of the configuration attributes and operations available for Parlay X 3.0 Payment/Diameter network protocol plug-in instances.

To see a	Refer to
Detailed list of necessary for managing and configuring a plug-in instance	Properties for Parlay X 3.0 Payment/Diameter
Configuration workflow	Configuration Workflow for Parlay X 3.0 Payment/Diameter

To see a	Refer to
List of operations and attributes related to management and provisioning	Provisioning Workflow for Parlay X 3.0 Payment/Diameter
Reference of management attributes and operations	Reference: Attributes and Operations for Parlay X 3.0 Payment/Diameter

Properties for Parlay X 3.0 Payment/Diameter

Property	Description
Managed object in Administration Console	<domain name>–>OCSG–><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.payment.diameter.management.PaymentMBean
Network protocol plug-in service ID	Plugin_px30_payment_diameter
Network protocol plug-in instance ID	The ID is given when the plug-in instance is created, see Managing and Configuring the Plug-in Manager.
Supported Address Scheme	tel
North interfaces	com.bea.wlcp.wlng.px30.plugin.AmountChargingPlugin com.bea.wlcp.wlng.px30.plugin.ReserveAmountChargingPlugin
Service type	Payment
Exposes to the service communication layer a Java representation of:	Parlay X 3.0 Part 6: Payment

Property	Description
Interfaces with the network nodes using:	Diameter RFC3588 and RFC 4006
Deployment artifacts NT EAR wlng_nt_payment_px30.ear	Plugin_px30_payment_diameter.jar and px30_payment_service.jar
AT EAR: Normal wlng_at_payment_px30.ear	rest_payment.war and px30_payment.war
AT EAR: SOAP Only wlng_at_payment_px30_soap.ear	px30_payment.war

Configuration Workflow for Parlay X 3.0 Payment/Diameter

Below is an outline for configuring the plug-in using the Oracle Communications Services 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 3.0 Payment/Diameter.
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: Connected
   - Attribute: DestinationHost
   - Attribute: DestinationPort
   - Attribute: DestinationRealm
   - Attribute: OriginHost
   - Attribute: OriginPort

  - Attribute: OriginRealm

4. Use Operation: connect to connect to the Diameter server.
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 3.0 Payment/Diameter.
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.

Management of Parlay X 3.0 Payment/Diameter

The Parlay X 3.0 Payment/Diameter plug-in instance can be explicitly connected to the Diameter server. It does not connect to the server by default. The service has a connection status that will be preserved after service redeployment and server restart.

Use:

- Operation: connect
- Operation: disconnect

Use Operation: connect after any changes to the configuration attributes. Changes does not take affect until this operation is invoked.

Provisioning Workflow for Parlay X 3.0 Payment/Diameter

No provisioning operations.

Reference: Attributes and Operations for Parlay X 3.0 Payment/Diameter

Managed object: Communication Services-><plug-in instance ID>

MBean: com.bea.wlcp.wlng.plugin.payment.diameter.management.PaymentMBean

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

- Attribute: Connected

- Attribute: DestinationHost
- Attribute: DestinationPort
- Attribute: DestinationRealm
- Attribute: OriginPort
- Attribute: OriginHost
- Attribute: OriginRealm
- Operation: connect
- Operation: disconnect

Attribute: Connected (r)

Read-only.

Scope: Server

Format: Boolean

Unit: n/a

Displays the status of the connection to the Diameter server.

Displays:

- `true`, if connected.

`false`, if not connected.

Attribute: Connected

Scope: Cluster

Unit: n/a

Format: String

Specifies the value of the Destination-Host AVP in Diameter requests.

Example:

host.destination.com

Attribute: DestinationHost

Scope: Cluster

Unit: n/a

Format: String

Specifies the host name of the Diameter server to connect to.

Example:

host.destination.com

Attribute: DestinationPort

Scope: Cluster

Unit: n/a

Format: int

Specifies the port on the Diameter server to connect to.

Example:

3588

Attribute: DestinationRealm

Scope: Cluster

Unit: n/a

Format: String

Specifies the Destination-Realm AVP in Diameter requests.

Example:

destination.com

Attribute: OriginHost

Scope: Cluster

Unit: n/a

Format: String

Specifies the Origin-Host AVP in Diameter requests.

Example:

host.oracle.com

Attribute: OriginPort

Scope: Cluster

Unit: n/a

Format: int

Specifies the local originator port to be used for the connection to the Diameter server.

Example:

7002

Attribute: OriginRealm

Scope: Cluster

Unit: n/a

Format: String

Specifies the Origin-Realm AVP in Diameter requests.

Example:

oracle.com

Operation: connect

Scope: Cluster

Connects to the Diameter server.

Once connected, the service will try to reconnect to the Diameter server if the server is restarted or the plug-in is redeployed.

Signature:

`connect()`

Table 25-1 connect

connect	
Parameter	**Description**
-	-

Operation: disconnect

Scope: Cluster

Disconnects from the Diameter server.

Once disconnected, the plug-in will not try to reconnect to the DAMETER server if the server is restarted or the plug-in is redeployed.

Signature:

```
disconnect()
```

Table 25-2 disconnect

disconnect	
Parameter	**Description**
-	-

CHAPTER 26

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	Properties for Parlay X 2.1 Terminal Location/MLP
Configuration workflow	Configuration Workflow for Parlay X 2.1 Terminal Location/MLP

To see a	Refer to
List of operations and attributes related to management and provisioning	Provisioning Workflow for Parlay X 2.1 Terminal Location/MLP Management Operations for Parlay X 2.1 Terminal Location/MLP
Reference of management attributes and operations	Reference: Attributes and Operations for Parlay X 2.1 Terminal Location/MLP

Properties for Parlay X 2.1 Terminal Location/MLP

Property	Description
Managed object in Administration Console	<domain name>–>OCSG–><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 Managing and Configuring the Plug-in Manager.
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 NT EAR wlng_nt_terminal_location_px21.ear	Plugin_px21_terminal_location_mlp.jar, px21_terminal_location_service.jar, and terminal_location_mlp.war
AT EAR: Normal wlng_at_terminal_location_px21.ear	px21_terminal_location.war, px21_terminal_location_callback.jar, and rest_terminal_location.war
AT EAR: SOAP Only wlng_at_terminal_location_px21_soap.ear	px21_terminal_location.war and px21_terminal_location_callback.jar

Configuration Workflow for Parlay X 2.1 Terminal Location/MLP

Below is an outline for configuring the plug-in instance using the Oracle Communications Services 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: DecimalDegreesToDMSH
- 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: DecimalDegreesToDMSH
- 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 Oracle Communications Services 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 Oracle Communications Services 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 Oracle Communications Services Gatekeeper service ID. If set to an empty string, the `<serviceid>` element 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: DecimalDegreesToDMSH

Scope: Cluster

Unit: n/a

Format: Boolean.

Specifies if the coordinates provided by an application, in the form of decimal degrees, shall be converted to DMSH (Degrees Minutes Seconds Hemisphere) format.

Enter:

- true, to convert to SMSH
- false, to use decimal degrees

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 that this value.

If the duration is not provided in the request, this value is used.

CHAPTER 27

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

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	Properties for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction
Configuration workflow	Configuration Workflow for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction
List of operations and attributes related to management and provisioning	None.
Reference of management attributes and operations	Reference: Attributes and Operations for Parlay X 3.0 Audio Call/Parlay MultiParty Call Control and Call User Interaction

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>–>OCSG–><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 Oracle Communications Services 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: 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: RetensionTime

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.

CHAPTER 28

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 Oracle Communications Services 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
- Notifications Cache

To see a	Refer to
Detailed list of necessary for managing and configuring the plug-in instance	Properties for Parlay X 2.1 Presence/SIP
Configuration workflow	Configuration Workflow for Parlay X 2.1 Presence/SIP
List of operations and attributes related to management and provisioning	Provisioning Workflow for Parlay X 2.1 Presence/SIP Management Operations for Parlay X 2.1 Presence/SIP
Reference of management attributes and operations	Reference: Attributes and Operations for Parlay X 2.1 Presence/SIP

URI Cache

In Parlay X 2.1 Presence, the SIP URI of the user (the presentity in PresenceSupplier cases, the watcher in PresenceConsumer cases) is not passed as an argument. Instead Oracle Communications Services Gatekeeper maps the user URI to the application instance ID of the user application, 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.

Note: In the case of the PresenceSupplier interface, the application can override the default mapping by including a tunneled parameter in the header of its request.

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>–>OCSG–><server name>–>Communication Services–>Plugin_px21_presence_sip
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName=Plugin_px21_presence_sip Type=com.bea.wlcp.wlng.plugin.presence.sip.management.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

Property	Description
Exposes to the service communication layer a Java representation of:	Parlay X 2.1 Part 14: Presence
Interfaces with the network nodes using:	SIP: Session Initiation Protocol, RFC 3261.
Deployment artifacts NT EAR wlng_nt_presence_px21.ear	Plugin_px21_presence_sip.jar, px21_presence_service.jar, and px21_presence_sip.jar
AT EAR: Normal wlng_at_presence_px21.ear	px21_presence.war, px21_presence_callback.jar, and rest_presence.war
AT EAR: SOAP Only wlng_at_presence_px21_soap.ear	px21_presence.war and px21_presence_callback.jar

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: SubscriptionCleanupTimerValue

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 Presence/SIP.
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.

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: listNotificationsCache
- Operation: listSubscriptionsCache
- Operation: listURImappingCache
- Operation: updateSubscriptionToBeconfirmed

Reference: Attributes and Operations for Parlay X 2.1 Presence/SIP

Managed object: Communication Services->Plugin_presence_sip

MBean: com.bea.wlcp.wlng.plugin.presence.sip.impl.PresenceMBean

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

- Attribute: DefaultNotificationCount

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: 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 address of the presence server or another IMS node that proxies the request.

Attribute: PresenceXDMSAddress

Scope: Cluster

Unit: n/a

Format: String formatted as a SIP URI

Specifies the XCAP root URI of the XDM server.

Attribute: PresenceXDMSPresrulesPostfix

Scope: Cluster

Unit: n/a

Format: String formatted as a partial path.

Specifies the last-part of the XCAP Document selector for presence rules. This part is after the XCAP User Identifier(XUI). Generally the selector URI should be the string of the following type: [XCAP_ROOT]/[AUID]/users/[XUI]/[document_name]

Example:

```
/presrules
```

Attribute: PresenceXDMSPresrulesPrefix

Scope: Cluster

Unit: n/a

Format: String formatted as a path

The pre-part of the XCAP Document selector for presence rules. This part is before the XCAP User Identifier(XUI). Generally the selector URI should be the string of the following type: [XCAP_ROOT]/[AUID]/users/[XUI]/[document_name]

Example:

```
/services/pres-rules/users/
```

Attribute: PresenceXDMSProviderClassName

Scope: Cluster

Unit: n/a

Format: Class name as string

The class name of XDM Server provider. This is a pluggable function to allow third party vendors of XDMS to customize their own XCAP client. This class must implement the "com.bea.wlcp.wlng.plugin.presence.sip.south.xcap.XCAPClient" interface.

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: 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 28-1 clearCache

clearCache	
Parameter	**Description**
cacheToClear	Name of the cache to clear. Valid options: • NOTIFICATIONS -clears the notification cache • SUBSCRIPTIONS -clears the subscriptions cache • URIMAPPINGS -clears the URI mappings cache • ALL -clears all caches.

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 28-2 getApplicationInstance

getApplicationInstanceGroup	
Parameter	**Description**
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 28-3 getApplicationInstanceSIPURI

getApplicationInstanceSIPURI	
Parameter	**Description**
ApplicationInstanceID	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 28-4 listNotificationsCache

listNotificationsCache	
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 28-5 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 28-6 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 28-7 removeApplicationInstanceFromCache

removeApplicationInstanceFromCache	
Parameter	**Description**
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 28-8 removeNotification

removeNotification	
Parameter	**Description**
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 28-9 removeSubscription

removeSubscription	
Parameter	**Description**
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 28-10 setApplicationInstanceSIPURI

setApplicationInstanceSIPURI	
Parameter	**Description**
ApplicationInstanceID	ID of the application instance.
URI	SIP URI. For example: sip:name@somedomain.org

Operation: updateSubscriptionToBeconfirmed

Scope: Cluster

Updates the xml rule in XDMS to status: confirm.

Forces a blocked subscription to be pending/confirmed.

Signature:

```
updateSubscriptionToBeconfirmed(presentity: String, watcher: String)
```

Table 28-11 setApplicationInstanceSIPURI

setApplicationInstanceSIPURI	
Parameter	**Description**
Presentity	ID of the application instance.
Watcher	SIP URI. For example: sip:name@somedomain.org

CHAPTER 29

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	Properties for Extended Web Services Subscriber Profile/LDAPv3
Configuration workflow	Configuration Workflow for Extended Web Services Subscriber Profile/LDAPv3
List of operations and attributes related to management and provisioning	Management for Extended Web Services Subscriber Profile/LDAPv3 Provisioning for Extended Web Services Subscriber Profile/LDAPv3
Reference of management attributes and operations	Reference: Attributes and Operations for Extended Web Services Subscriber Profile/LDAPv3

Properties for Extended Web Services Subscriber Profile/LDAPv3

Property	Description
Managed object in Administration Console	<domain name>–>OCSG–><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 Managing and Configuring the Plug-in Manager.
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
Interfaces with the network nodes using:	LDAP
Deployment artifacts NT EAR wlng_nt_subscriber_profile_ews.ear	ews_subscriber_profile_service.jar and Plugin_ews_subscriber_profile_ldap.jar

Property	Description
AT EAR: Normal wlng_at_subscriber_profile_ews.ear	ews_subscriber_profile.war and rest_subscriber_profile.war
AT EAR: SOAP Only wlng_at_subscriber_profile_ews_soap.ear	ews_subscriber_profile.war

LDAP Server Schema

A schema is used for constructing queries, see Listing 29-1.

Listing 29-1 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 29-1 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 29-1 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 29-2 is an example of how a directory information tree is built up using the above schema.

Listing 29-2 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 29-2 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: • Missing or incorrect configuration. • LDAP server is unreachable. • Internal errors.

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 29-3 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 29-4 updateSchemaURL

updateSchemaURL	
Parameter	**Description**
SchemaURL	URL to the LDAP database schema. Examples: `file:///d:/ldap/schema.xml` (Windows systems) `file://ldap/schema.xml` (UNIX systems)

CHAPTER 30

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

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	Properties for Extended Web Services WAP Push/PAP
Configuration workflow	Configuration Workflow for Extended Web Services WAP Push/PAP

To see a	Refer to
List of operations and attributes related to management and provisioning	Management for Extended Web Services WAP Push/PAP Provisioning for Extended Web Services WAP Push/PAP
Reference of management attributes and operations	Reference: Attributes and Operations for WAP Push/PAP

Properties for Extended Web Services WAP Push/PAP

Property	Description
Managed object in Administration Console	<domain name>–>OCSG–><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.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 Managing and Configuring the Plug-in Manager.
Supported Address Scheme	tel wapuser
North interface	com.bea.wlcp.wlng.ews.plugin.PushMessagePlugin com.bea.wlcp.wlng.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 NT EAR wlng_nt_push_message_ews.ear	ews_push_message_service.jar, Plugin_ews_push_message_pap.jar, and ews_push_message_pap.war
AT EAR: Normal wlng_at_push_message_ews.ear	ews_push_message.war, ews_push_message_callback.jar, and rest_push_message.war
AT EAR: SOAP Only wlng_at_push_message_ews_soap.ear	ews_push_message.war and ews_push_message_callback.jar

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.

CHAPTER 31

Managing and Configuring Native MM7 Communication Services

The following section describes configuration and maintenance attributes and operations for the communication service that exposes Native MM7 interfaces. The sections also provide a workflow for the configuration:

Native MM7

This section contains a description of the configuration attributes and operations available for Native MM7 network protocol plug-in instances.

To see a	Refer to
Detailed list of necessary for managing and configuring a plug-in instance	Properties for Native MM7
Configuration workflow	Configuration Workflow for Native MM7
List of operations and attributes related to management and provisioning	Provisioning Workflow for Native MM7
Reference of management attributes and operations	Reference: Attributes and Operations for Native MM7

Properties for Native MM7

Property	Description
Managed object in Administration Console	<domain name>–>OCSG–><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.mm7.management.Mm7MBean
Network protocol plug-in service ID	Plugin_multimedia_messaging_mm7
Network protocol plug-in instance ID	The ID is given when the plug-in instance is created, see Managing and Configuring the Plug-in Manager.
Supported Address Scheme	tel mailto short
North interfaces	com.bea.wlcp.wlng.mm7.plugin.MmsPlugin com.bea.wlcp.wlng.mm7.callback.MmsVaspCallback
Service type	Mm7
Exposes to the service communication layer a Java representation of:	3GPP TS 23.140 V5.3.0 (REL-5-MM7-1-2.xsd)

Property	Description
Interfaces with the network nodes using:	MM7 (REL-5-MM7-1-0 or REL-5-MM7-1-2)
Deployment artifacts	Plugin_multimedia_messaging_mm7.jar mm7_service.jar 1_0_mm7_vasp.war 1_2_mm7_vasp.war packaged in wlng_nt_multimedia_messaging_mm7.ear mm7.war mm7_callback_client.jar, packaged in wlng_at_multimedia_messaging_mm7.ear

Configuration Workflow for Native MM7

Below is an outline for configuring the plug-in using the Oracle Communications Services 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 Native 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: Mm7RelayServerAddress
   - Attribute: HTTPBasicAuthentication: if using HTTP basic authentication also define:
     - Attribute: HTTPBasicAuthenticationUsername
     - Attribute: HTTPBasicAuthenticationPassword
   - 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 Native 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:

- For REL-5-MM7-1-0 the default URL is `http://<IP Address of NT server>:<port>/1_0_mm7_vasp/mms_vasp`
- For REL-5-MM7-1-2 the default URL is `http://<IP Address of NT server>:<port>/1_2_mm7_vasp/mms_vasp`

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 Native 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 Oracle Communications Services Gatekeeper for polling, use Operation: addVASIDMapping. Use the following operations to manage the offline registrations:
   - Operation: listAllVASIDMapping
   - Operation: addVASPIDMapping
   - Operation: removeReceiveMmsNotification
2. To register online notifications, that is, to manage registrations for mobile originated messages on behalf of an application, use: Operation: removeVASIDMapping. Use the following operations to manage the online registrations:
   - Operation: listAllVASPIDMapping
   - Operation: enableReceiveMmsNotification
   - Operation: removeStatusReporting

Reference: Attributes and Operations for Native MM7

Managed object: Communication Services-><plug-in instance ID>

MBean:
com.bea.wlcp.wlng.plugin.multimediamessaging.mm7.management.MessagingManagementMBean

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

- Attribute: HTTPBasicAuthentication
- Attribute: HTTPBasicAuthenticationUsername
- Attribute: HTTPBasicAuthenticationPassword
- Attribute: XSDVersion
- Operation: addVASIDMapping
- Operation: addVASPIDMapping
- Operation: enableReceiveMmsNotification
- Operation: listAllVASIDMapping
- Operation: listAllVASPIDMapping
- Operation: removeReceiveMmsNotification
- Operation: removeStatusReporting
- Operation: removeVASIDMapping

Attribute: Mm7RelayServerAddress

Scope: Cluster

Unit: n/a

Format: String

Specifies the address of the MM7 Relay Server. The address is an HTTP URL.

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. This is equivalent to the Application Instance ID.

Attribute: HTTPBasicAuthenticationPassword

Scope: Cluster

Unit: n/a

Format: String

The password to use for HTTP basic authentication towards the MM7 server.

Attribute: XSDVersion

Scope: Cluster

Unit: n/a

Format: String [REL-5-MM7-1-0, REL-5-MM7-1-2]

Specifies the 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 the 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: addVASIDMapping

Scope: Cluster

Adds the service provider VAS ID and the Oracle Communications Services Gatekeeper VAS ID mapping for use with the submit request.

Signature:

```
addVASIDMapping(spVasID: String, wlngVasID: String)
```

Table 31-1 addVASIDMapping

addVASIDMapping	
Parameter	**Description**
spVasID	The service provider's VAS ID.
wlngVasID	The VAS ID that is to be sent to the MMSC.

Operation: addVASPIDMapping

Scope: Cluster

Adds the service provider VASP ID and the Oracle Communications Services Gatekeeper VASP ID mapping for use with the `SubmitReq` operation.

Signature:

```
addVASPIDMapping(spVaspID: String, wlngVaspID: String)
```

Table 31-2 addVASPIDMapping

addVASPIDMapping	
Parameter	**Description**
spVaspID	The service provider's VASP ID.
wlngVaspID	The VASP ID that is to be sent to the MMSC.

Operation: enableReceiveMmsNotification

Scope: Cluster

Sets up delivery notification for network-triggered message delivery. Messages matching this configuration result in a callback to the application via the `DeliverReq` operation.

Signature:

```
enableReceiveMmsNotification(Address: String, appInstGroupID: String,
applicationURI: String)
```

Table 31-3 enableReceiveMmsNotification

enableReceiveMmsNotification	
Parameter	**Description**
Address	The destination address of the MMS
appInstGroupID	The application instance group ID associated with this notification
applicationURI	The URI where the application can be reached

Operation: enableStatusReporting

Scope: Cluster

Sets up status report notification for delivery status and read-reply status for application-initiated messages.

Signature:

```
enableStatusReporting(appInstGroupID: String, applicationURI: String)
```

Table 31-4 enableStatusReporting

enableStatusReporting	
Parameter	**Description**
appInstGroupID	The application instance group ID associated with this notification
applicationURI	The URI where the application can be reached

Operation: getReceiveMmsNotificationForAddress

Scope: Cluster

Checks to see if a notification is already set up for this exact address. If no entry is returned, the address can be used to set up a new notification

Signature:

```
getReceiveMmsNotificationForAddress(Address: String)
```

Table 31-5 getReceiveMmsNotificationForAddress

getReceiveMmsNotifcation ForAddress	
Parameter	**Description**
Address	The destination address of the MMS

Operation: getReceiveMmsNotificationMatches

Scope: Cluster

Searches existing configurations for Receive Mms notifications for an address pattern, using regular expressions. If there are no matching configurations, null is returned. This can be used to find out which application is registered for notifications for a specific address.

Note: The operation returns at most 1 match. If there are multiple entries, only the first will be returned, although in normal operation overlapping entries should not be possible.

Signature:

```
getReceiveMmsNotificationMatches(Address: String)
```

Table 31-6 getReceiveMmsNotificationMatches

getReceiveMmsNotifcation Matches	
Parameter	**Description**
Address	The destination address pattern of the MMS: for example, 1234 or .@oracle.com

Operation: listAllVASIDMapping

Scope: Cluster

Lists all VAS ID mappings

Signature:

`listAllVASIDMapping()`

Table 31-7 listAllVASIDMapping

listOfflineNotificationInfo	
Parameter	**Description**
-	-

Operation: listAllVASPIDMapping

Scope: Cluster

Lists all VASP ID mappings

Signature:

`listAllVASPIDMapping()`

Table 31-8 listALLVASPIDMapping

listALLVASPIDMapping	
Parameter	**Description**
-	-

Operation: listReceiveMmsNotifications

Scope: Cluster

Lists all established delivery notifications

Signature:

```
listReceiveMmsNotifications()
```

Table 31-9 listReceiveMmsNotifications

listReceiveMmsNotifications	
Parameter	**Description**
-	-

Operation: listStatusReportingNotifications

Scope: Cluster

Lists all established status report notifications

Signature:

```
listStatusReportingNotifications()
```

Table 31-10 listStatusReporting

listStatusReporting	
Parameter	**Description**
-	-

Operation: listVASIDMapping

Scope: Cluster

Lists the VAS ID mapping that corresponds to the specified spVasID

Signature:

```
listVASIDMapping(spVasID: String)
```

Table 31-11 listVASIDMapping

listVASIDMapping	
Parameter	**Description**
spVasID	The service provider VAS ID to be matched.

Operation: listVASPIDMapping

Scope: Cluster

Lists the VASP ID mapping that corresponds to the specified spVaspID

Signature:

```
listVASPIDMapping(spVaspID: String)
```

Table 31-12 listVASPIDMapping

listVASPIDMapping	
Parameter	**Description**
spVaspID	The service provider VASP ID to be matched.

Operation: removeReceiveMmsNotification

Scope: Cluster

Removes an existing delivery notification

Signature:

```
removeReceiveMmsNotification(address: String)
```

Table 31-13 removeReceiveMmsNotification

removeReceiveMmsNotification	
Parameter	**Description**
address	The destination address of the MMS

Operation: removeStatusReporting

Scope: Cluster

Removes an existing status report notification.

Signature:

```
removeStatusReporting(applicationInstanceGroupID: String)
```

Table 31-14 removeStatusReporting

removeStatusReporting	
Parameter	**Description**
applicationInstanceGroupID	The application instance group ID associated with the notification.

Operation: removeVASIDMapping

Scope: Cluster

Removes an established VAS ID mapping.

Signature:

```
removeVASIDMapping(spVasID: String)
```

Table 31-15 removeVASIDMapping

removeVASIDMapping	
Parameter	**Description**
spVasID	The service provider VAS ID whose mapping is to be removed

Operation: removeVASPIDMapping

Scope: Cluster

Removes an established VASP ID mapping.

Signature:

```
removeVASIPDMapping(spVasID: String)
```

Table 31-16 removeVASPIDMapping

removeVASPIDMapping	
Parameter	**Description**
spVaspID	The service provider VASP ID whose mapping is to be removed

CHAPTER 32

Managing and Configuring Native SMPP Communication Services

The following section describes configuration and maintenance attributes and operations for the communication service that exposes native SMPP interfaces. The sections also provide a workflow for the configuration:

- Native SMPP
- Properties for Native SMPP Facade
- Properties for Native SMPP Plug-in
- Configuration Workflow for Native SMPP Communication Service
- Provisioning Workflow for Native SMPP Communication Service
- Reference: Attributes and Operations for Native SMPP Facade
- Reference: Attributes and Operations for Native SMPP Plug-in

Native SMPP

This section contains a description of the configuration attributes and operations available for Native SMPP network protocol plug-in instances and the Native SMPP Facade.

Applications interact with the Native SMPP Facade and the Native SMPP Plug-in instances interact with an SMPP SMSC.

Unlike other Service Facades, the Native SMPP Service Facade is deployed in the Network Tier, so applications using the SMPP Native Communication Service need to be able to connect

directly to the Network Tier. The network and any firewall need to be configured to allow connecting to the ports defined for the Native SMPP Service Facade.

There is no fail-over between Network Tier servers, so in order to support high-availability features, redundant SMSCs and redundant network cards are required.

To optimize system utilization the application and the SMSC should load balance the request between all Network Tier servers.

In certain configurations, network-triggered messages for an application can arrive at a Network Tier server that does not have a bind to that application,. In this case, it is possible to turn on a feature that allows that server to place the message in a JMS queue from which a Network Tier server that does have a bind to the application can pick the message up and forward it on to the application. This is done by setting Attribute: OfflineMO of the Native SMPP Service Facade (sms_smpp_connector). Messages stay alive for a configurable length of time. After the time has elapsed, the messages are removed and an EDR is written.

Note: When running the Native SMPP communication service, you may receive the following exception from your NT servers.

```
ERROR com.bea.wlcp.wlng.legacy.smpp.connector.SmppChannelProcessor -
Could not find request for received response
```

This exception can be ignored

To see a	Refer to
Detailed list of necessary for managing and configuring the Native SMPP Facade	Properties for Native SMPP Facade
Detailed list of necessary for managing and configuring a plug-in instance	Properties for Native SMPP Plug-in
Configuration workflow	Configuration Workflow for Native SMPP Communication Service
List of operations and attributes related to management and provisioning	Provisioning Workflow for Native SMPP Communication Service

To see a	Refer to
Reference of management attributes and operations for the Native SMPP Facade	Reference: Attributes and Operations for Native SMPP Facade
Reference of management attributes and operations for the Native SMPP plug-in	Reference: Attributes and Operations for Native SMPP Plug-in

Properties for Native SMPP Facade

Property	Description
Managed object in Administration Console	<domain name>–>OCSG–><server name>–>Communication Services–>sms_smpp_connector
MBean	Domain=com.bea.wlcp.wlng Name=wlng_nt InstanceName=sms_smpp_connector Type=com.bea.wlcp.wlng.legacy.smpp.connector.management.LegacySmppMBean
Exposes this interface to applications	SMPP v3.4, Short Message Peer to Peer, Protocol Specification v3.4, Document Version:- 12-Oct-1999 Issue 1.2.
Deployment artifacts	Plugin_sms_smpp.jar and sms_smpp_connector.jar packaged in wlng_nt_sms_smpp.ear

Properties for Native SMPP Plug-in

Property	Description
Managed object in Administration Console	<domain name>–>OCSG–><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.legacy.smpp.management.SmppPluginMBeanImpl
Network protocol plug-in service ID	Plugin_sms_smpp
Network protocol plug-in instance ID	The ID is given when the plug-in instance is created, see Managing and Configuring the Plug-in Manager.
Supported Address Scheme	tel
North interfaces	com.bea.wlcp.wlng.smpp.callback.UpstreamSmppCallback com.bea.wlcp.wlng.smpp.plugin.DownstreamSmppPlugin
Service type	SMPP
Exposes to the service communication layer a Java representation of:	SMPP v 3.4
Interfaces with the network nodes using:	SMPP v 3.4
Deployment artifacts	See deployment artifacts for Native SMPP Faces, in Properties for Native SMPP Facade

Configuration Workflow for Native SMPP Communication Service

Below is an outline for configuring the Native SMPP Communication Service using the Oracle Communications Services Gatekeeper Administration Console:

1. Using the Management Console or an MBean browser, select the MBean for the Native SMPP Service Facade. Navigate to Communication Services–>sms_smpp_connector
2. Configure the behavior of the Service Facade:
   - Attribute: InactivityTimeOut
   - Attribute: InitiationTimeOut
   - Attribute: ServerAddress
   - Attribute: ServerPort
   - Attribute: SessionTimeOut
   - Attribute: SmscSystemId
   - Attribute: TransactionTimeOut
   - Attribute: OfflineMO
   - Attribute: LooseBinding
3. Apply the configuration settings for the Native SMPP Facade using Operation: updateAllListenerPorts.
4. 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 Native SMPP Plug-in.
5. 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.
6. Configure the behavior of the plug-in instance:
   - Attribute: BindType
   - Attribute: DeliverSmRespCommandStatus
   - Attribute: EnquireLinkRequestTimerValue

- Attribute: EnquireLinkTimerValue
- Attribute: EsmeAddressRange
- Attribute: EsmeNpi
- Attribute: EsmePassword
- Attribute: EsmeSystemId
- Attribute: EsmeSystemType
- Attribute: EsmeTon
- Attribute: LocalAddress
- Attribute: LocalPort
- Attribute: NumberOfConnections
- Attribute: EnableDeleteAfterQuery
- Attribute: EnableDeleteAfterNotify
- Attribute: EnableDeleteAfterCancel
- Attribute: RetryTimesBeforeGiveUp
- Attribute: RetryTimesBeforeReconnect
- Attribute: RequestTimerValue
- Attribute: SmscAddress
- Attribute: SmscPort

7. Apply the configuration settings for the Native SMPP Plug-in instance using Operation: resetConnections.
8. 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 Native SMPP Plug-in.
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 Native SMPP Communication Service

1. To register application instances to use the Native SMPP Communication Service use Operation: addApplicationSpecificSettings in the MBean for the Native SMPP Service Facade. Use the following operations to manage the account settings:
   - Operation: listApplicationSpecificSettings
   - Operation: deleteApplicationSpecificSettings
2. Apply the provisioning settings using Operation: updateAllListenerPorts.

Continue with the provisioning of service provider accounts and application accounts.

Reference: Attributes and Operations for Native SMPP Facade

Managed object: Communication Services->sms_smpp_connector

MBean: com.bea.wlcp.wlng.legacy.smpp.connector.management.LegacySmppMBean

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

- Attribute: ServerPort
- Attribute: ServerAddress
- Attribute: OfflineMO
- Attribute: LooseBinding
- Attribute: SmscSystemId
- Attribute: InitiationTimeOut
- Attribute: TransactionTimeOut
- Attribute: InactivityTimeOut
- Attribute: SessionTimeOut
- Operation: addApplicationSpecificSettings
- Operation: deleteApplicationSpecificSettings

- Operation: listApplicationSpecificSettings
- Operation: listListenerPortsStatus
- Operation: listOpenedPorts
- Operation: stopPort
- Operation: updateAllListenerPorts

Attribute: ServerPort

Scope: Server

Unit: n/a

Format: Int

Specifies the default port applications use to connect to the Native SMPP Communication Service. Specific ports can be defined using Operation: addApplicationSpecificSettings.

Operation: updateAllListenerPorts must be invoked after this operation in order to update the listener port.

Attribute: ServerAddress

Scope: Server

Unit: n/a

Format: String

Specifies the server address applications use to connect to the Native SMPP Communication Service. The address can be expressed as an IP-address or as a host name. The address or host name must resolve to a local address.

Attribute: OfflineMO

Scope: Server

Unit: n/a

Format: Boolean

Specifies whether or not the JMS-based routing functionality for network-triggered messages is turned on. If it is on, a message that comes in from the network to an NT server that does not have

an active bind to the appropriate application can be placed in a JMS queue from which another server that does have an active bind can fetch it and send it to the application. The default is false.

The time that the message stays alive in the JMS queue is configurable. The default value is 3600000 milliseconds. To change this value, in the administrative console:

1. Select **Services** ->**Messaging**->**JMS Modules**.
2. Click on **WLNGJMSResource**. The Settings page opens.
3. On the **Configuration** tab, click **LegacySMSConnectionFactory**. The **Settings for LegacySMSConnectionFactory** page opens.
4. On the **Configuration** tab, select the **Default Delivery** sub-tab.
5. Make your changes to the **Default Time-to-Live** attribute.
6. Click the **Save** button.
7. Click the **Activate Changes** button in the **Change Center**.

Attribute: LooseBinding

Scope: Server

Unit: n/a

Format: Boolean

Controls what will happen on application binds/unbinds.

If true, the following applies:

As long as there are transmitting capable connections (TX, TRX) setup from applications, TX and TRX connections will be kept open to SMSCs.

As long as there are receiving capable connections (RX, TRX) setup from applications, RX and TRX connections will be kept open to SMSCs.

If false, the binding rules are more strict:

As long as there are TX connections setup from applications, TX connections will be kept open to SMSCs.

As long as there are RX connections setup from applications, RX connections will be kept open to SMSCs.

As long as there are TRX connections setup from applications, TRX connections will be kept open to SMSCs

The default is true.

Attribute: SmscSystemId

Scope: Cluster

Unit: n/a

Format: String

Specifies the SMSC system ID. This is sent to an application upon a successful BIND request from the application.

Attribute: InitiationTimeOut

Scope: Cluster

Unit: Seconds

Format: Int

Specifies the maximum time-out value for the time period between an application's network connection setup request and BIND request.

If the time-out value is reached, the connection is terminated.

Use 0 (zero) for no time-out.

Attribute: TransactionTimeOut

Scope: Cluster

Unit: Seconds

Format: Int

Specifies the maximum time-out period allowed between the Native SMPP service sends a request to an application and a response to the request is received.

If the time-out value is reached, the connection is terminated.

Use 0 (zero) for no time-out.

Attribute: InactivityTimeOut

Scope: Cluster

Unit: Seconds

Format: Int

Specifies the maximum time-out period allowed for inactivity between an application's request towards the Native SMPP service.

If the time-out value is reached, the connection is terminated.

Use 0 (zero) for no time-out.

Attribute: SessionTimeOut

Scope: Cluster

Unit: Seconds

Format: Int

Specifies the minimum time-out period between the Native SMPP service's sending of ENQUIRE LINK requests to an application.

Use 0 (zero) for no time-out.

Operation: addApplicationSpecificSettings

Scope: Cluster

Specifies connection details for an application with a given application instance ID.

Signature:

```
addApplicationSpecificSettings(applicationInstanceGroupId: String,
acceptPort: int, maxSession: int, subsequentOperationsAllowed: Boolean,
notificationEnabled: Boolean, addressRange: String)
```

Table 32-1 addApplicationSpecificSettings

addApplicationSpecificSettings	
Parameter	**Description**
applicationInstanceGroupId	ID of the application instance group the settings are valid for.
acceptPort	The port that the application is allowed to bind to. Use negative value to allow any port. **Note:** Operation: updateAllListenerPorts must be invoked after this operation in order to update the listener port if a specific port (non-negative value) is given.
maxSession	The number of concurrent sessions the application is allowed to establish. Use negative value to allow any number of concurrent sessions.
subsequentOperationsAllowed	Specifies if the application is allowed to perform operations on a previously sent short message. The subsequent operations are: • querySm • replaceSm • cancelSm Enter: • **true** to allow • **false** to deny Setting this to false reduces the resource utilization by the Native SMPP Communication Service since it does not need to track each request in its Store.

Table 32-1 addApplicationSpecificSettings

addApplicationSpecificSettings	
Parameter	**Description**
notificationEnabled	Specifies if the application is allowed to receive network-triggered messages or not. If allowed, it can send the operations bind_tranceiver and bind_receiver. Enter: Enter: • **true** to allow • **false** to deny
addressRange	Specifies the address range to listen for network-triggered short messages. Only messages that are sent to this address range will be forwarded to the application. The address range is expressed using regular expressions. When binding a receiver or transceiver, the address_range in the bind operation must be equal to this value. Otherwise the bind will be rejected.See Appendix A in *SMPP Protocol Specification v3.4*. This setting is only valid if the application is allowed to receive network-triggered messages. Example: ^1234

Operation: deleteApplicationSpecificSettings

Scope: Cluster

Deletes connection details for an application with a given application instance ID. When removed, the application has no longer access to the Native SMPP Service.

Operation: updateAllListenerPorts must be invoked after this operation in order to update the listener port.

Signature:

```
deleteApplicationSpecificSettings(applicationInstanceGroupId: String)
```

Table 32-2 deleteApplicationSpecificSettings

deleteApplicationSpecificSettings	
Parameter	**Description**
applicationInstanceGroupId	ID of the application instance group to delete settings for.

Operation: listApplicationSpecificSettings

Scope: Cluster

Displays a list of the application instance IDs that have connection details specified.

Signature:

```
listApplicationSpecificSettings()
```

Table 32-3 listApplicationSpecificSettings

listApplicationSpecificSettings	
Parameter	**Description**
-	-

Operation: listListenerPortsStatus

Scope: Cluster

Displays a list of all ports that the Native SMPP Facade listens to and their status.

The format of the list is:

```
Port: <Port number> Status: <Status indicator>
```

Where <Status indicator> is one of the following:

- waiting for start

- normal
- waiting for remove

Signature:

```
listListenerPortStatus()
```

Table 32-4 listListenerPortStatus

listListenerPortStatus	
Parameter	**Description**
-	-

Operation: listOpenedPorts

Scope: Cluster

Displays a list of all ports that the Native SMPP Facade listens to. A list of all port in status: normal. See Operation: listListenerPortsStatus.

Signature:

```
listOpenedPorts()
```

Table 32-5 listOpenedPorts

listOpenedPorts	
Parameter	**Description**
-	-

Operation: stopPort

Scope: Cluster

Stops the Native SMPP Facade from listening to a given port. The Native SMPP Facade must previously have opened the port for listening.

Signature:

```
stopPort(port: int)
```

Table 32-6 stopPort

stopPort	
Parameter	**Description**
port	The port number to stop listening on.

Operation: updateAllListenerPorts

Scope: Cluster

Creates all server sockets based on the current configuration for all registered applications, see Operation: listApplicationSpecificSettings.

This operation must be invoked after any of these operations have been performed:

- Operation: addApplicationSpecificSettings
- Operation: deleteApplicationSpecificSettings

It must also be invoked after any changes to Attribute: ServerPort

Signature:

```
updateAllListenerPorts()
```

Table 32-7 updateAllListenerPorts

updateAllListenerPorts	
Parameter	**Description**
-	-

Reference: Attributes and Operations for Native SMPP Plug-in

Managed object: Communication Services-><plug-in instance ID>

MBean: com.bea.wlcp.wlng.plugin.legacy.smpp.management.SmppPluginMBeanImpl

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

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 [1, 3]

Specifies how the plug-in shall bind to the SMSC.

Use:

- **1** to bind as Transceiver
- **2** to bind as Transmitter
- **3** to bind as Receiver

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

Attribute: DeliverSmRespCommandStatus

Scope: Server

Unit: n/a

Format: int

Specifies how the plug-in shall respond to an SMSC if a network triggered short message cannot be delivered to the application that subscribed for notifications on incoming short messages.

It specifies the error code to used in the command_status field, see section *5.1.3 command_status* in *SMPP Protocol Specification v3.4*.

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

Attribute: EnquireLinkRequestTimerValue

Scope: Server

Unit: milliseconds

Format: int

Specifies the value of the timer used when sending Enquire Link requests.

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

Attribute: EnquireLinkTimerValue

Scope: Server

Unit: seconds

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.

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

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.

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

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

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

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.

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

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.

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

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

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

Attribute: LocalAddress

Scope: Server

Unit: n/a

Format: String

Specifies the local server address used when connecting to the SMPP SMSC. The address can be expressed as an IP-address or as a host name. The address or host name must resolve to a local address.

Enter `default` to not specify an address implicitly, but to use the default address of the server.

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

Attribute: LocalPort

Scope: Server

Unit: n/a

Format: Integer [1-65535]

Specifies the local port the plug-in uses when connecting to the SMPP SMSC.

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

Attribute: NumberOfConnections

Scope: Server

Unit: n/a

Format: Integer

Specifies the maximum number of connections to use towards the SMPP SMSC.

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

Attribute: EnableDeleteAfterQuery

Specifies whether to delete SMPP session information from storage after the receipt of query response with final message state.

The final message states contains all states except Enroute(=1)

Attribute: EnableDeleteAfterNotify

Specifies whether to deleteSMPP session information from storage after receipt of the delivery report with final message state.

The final message states contains all states except Enroute(=1)

Attribute: EnableDeleteAfterCancel

Specifies whether to cancel SMPP session information from storage after receipt of the cancel sm response.

Attribute: RetryTimesBeforeGiveUp

Scope: Server

Unit: n/a

Format: int

Specifies the maximum number of times to try to reconnect to the SMPP SMSC.

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

Attribute: RetryTimesBeforeReconnect

Scope: Server

Unit: n/a

Format: int

Specifies the maximum number of times to try to connect to the SMPP SMSC before a reconnect is attempted.

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

Attribute: RequestTimerValue

Scope: Server

Unit: milliseconds

Format: int

Specifies the value of the timer used when sending requests to the SMSC.

The setting will not be applied in until the plug-in service is restarted or Operation: resetConnections is performed.

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 until the plug-in service is restarted or Operation: resetConnections 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: resetConnections is performed.

Operation: listConnectionsStatus

Scope: Cluster

Displays a list of the status of each connection with the SMSC.

The information returned is either:

```
Connections status: [deactivated]
```

or

```
Connections status: [activated]
   bindType: [<Bind type>]
   Configured connections: [<number of connections>]
   bind links : [ <number of bound links> ]
```

<Bind Type> is one of:

- BIND_TRANCEIVER
- BIND_RECEIVER
- BIND_TRANSMITTER

Signature:

```
listConnectionsStatus()
```

Table 32-8 listConnectionsStatus

listConnectionsStatus	
Parameter	**Description**
-	-

Operation: resetConnections

Scope: Cluster

Resets the connections to the SMSC.

This operation needs to be invoked in order for the settings in the configuration attributes of the plug-in instance to take affect.

Signature:

```
resetConnections()
```

Table 32-9 resetConnections

resetConnections	
Parameter	**Description**
-	-

CHAPTER 33

SOAP2SOAP Communication Services

The following section describes how to generate deploy and configure SOAP2SOAP Communication Services. These Communication Services can be created using the Platform Development Studio or directly from the Administration Console:

About SOAP2SOAP Communication Services

SOAP2SOAP communication services are generated using the Platform Development Studio or directly from the Oracle Communications Services Gatekeeper Administration console.

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 Oracle Communications Services 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://<IP Address of AT server>:<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.

If the number of HTTP requests that time-out during a certain time-period exceeds the maximum allowed, the plug-in instance is taken out of operation for a configurable time-period. The HTTP time-out behavior is valid for request between the plug-in instance and the network node. The purpose of this feature is to prevent network nodes that are malfunctioning from incoming request.

Generating and Deploying SOAP2SOAP Communication Services

SOAP2SOAP Communication Services can be generated and deployed using the Administration console:

- Generate a Communication Service
- Deploy a Communication Service

See Generated Artifacts for the Communication Service for a description of the generated artifacts.

Generate a Communication Service

To generate a SOAP2SOAP Communication Service from the Administration console, open the SOAP-SOAP Generation page by selecting OCSG–>SOAP-SOAP from the Domain Structure group and enter configuration properties for the Communication Service according to Table 33-1.

SOAP2SOAP Communication Services can also be generated using the Eclipse wizard provided with the Platform Development Studio. See section Using the Eclipse Wizard in *Platform Development Studio - Developer's Guide*.

Table 33-1 SOAP-SOAP Communication Service generation fields

Field	Description
Name	Identifier that ties together a collection of Web Services. Will be a part of the names of the generated war and jar files and the package names for the for the Communication Service. Below is a list of the generated EARs and the WARs and JARs: wlng_at_<Communication service name>.ear: • <Name>.war • <Name>_callback.jar wlng_nt_<Communication service name>.ear: • <Name>.war • <Name>_service.jar • <Name>_soap_plugin.jar
Version	The version to be used in Used in META-INF/MANIFEST.MF in the EARs for the Communication Service.
ServiceType	Defines the service type. Used in SLAs, EDRs, statistics, etc. The service type to use in the SLAs is: com.bea.wlcp.wlng.soap2soap.<ServiceType>.plugin.<interface name from WSDL> Example: SmsServiceType
ContextPath	The context path for the Web Service. Example: myService

Table 33-1 SOAP-SOAP Communication Service generation fields

Field	Description
Service WSDLs	Enter the name of and path to each WSDL file that includes the service definition to be implemented by the new Communication Service. Use one file per line. The files must reside on a file system that can be reached by the Administration server. Example: D:\standards\ParlayX2.1\wsdl\parlayx_sms_send_service_2_2.wsdl D:\standards\ParlayX2.1\wsdl\parlayx_sms_notification_manager_service_2_3.wsdl D:\standards\ParlayX2.1\wsdl\parlayx_sms_receive_service_2_2.wsdl
Callback WSDLs	Enter the name of and path to each WSDL file that includes the callback service definition to be used by the new Communication Service in sending information to the service provider's application. Use one file per line. The files must reside on a file system that can be reached by the Administration server. Example: D:\standards\ParlayX2.1\wsdl\parlayx_sms_notification_service_2_2.wsdl

On successful generation of the Communication Service, the deployment screen is displayed, see Deploy a Communication Service.

Deploy a Communication Service

When a SOAP2SOAP Communication Service has been generated using the Administration Console, see Generate a Communication Service the deployment screen is displayed.

When deploying directly from the Administrating console, fill in the entry fields described in Table 33-2 and click the button **Deploy SOAP2SOAP**. In this case, the Communication Service is deployed to the same domain as where it was generated.

Table 33-2 SOAP-SOAP Communication Service deployment fields

Field	Description
Name	Name of the Communication Service. Same as the name entered in the Name field in SOAP2SOAP Communication Service Screen. Do not change this.
Version	The version use when deploying the Communication Services.
UserName	User name for an Oracle Communications Services Gatekeeper administrative user.
Password	Password that corresponds to the user name.

The generated Communication Service can also be deployed using WebLogic tools, see Generated Artifacts for the Communication Service.

Once the Communication Service is deployed, create one or more plug-in instances, see Managing and Configuring SOAP2SOAP Communication Services.

Generated Artifacts for the Communication Service

When generating a SOAP2SOAP Communication Service the following directory structure is created in the directory `$DOAMIN_HOME/soap2soap` on the Administration Server.

Listing 33-1 Directory structure when generating SOAP2SOAP Communication Services

```
| +- <Communication service name>_<version>/
| | | +- build.properties
| | +- build.xml
| | +- common.xml
| | +- <Communication service name>/
| | | | +- common/
```

```
| | | | | +- resources/
| | | | | | +- enabler/
| | | | | | +- facade/
| | | | | | +- handlerconfig.xml
| | | | +- dist/
| | | | | +- com.bea.wlcp.wlng.soap2soap.<service type>.store_<version>.jar
| | | | | +- wlng_at_<Communication service name>.ear
| | | | | +- wlng_nt_<Communication service name>.ear
| | | | +- plugins/
| | | | | +- soap/
| | | | | +- tmp/
```

All Java source files, build files, configuration files, and deployable artefacts are created under the directory $DOMAIN_HOME/soap2soap. The source files are there mainly for reference, and the only files that are necessary to deploy the Communication Service are:

- `wlng_at_<Communication service name>.ear`
- `wlng_nt_<Communication service name>.ear`

wlng_at_<Communication service name>.ear should be deployed in the Access Tier server.

wlng_nt_<Communication service name>.ear should be deployed in the Network Tier server.

The generated Communication Service can be deployed as a part of the generation process, as described above or using standard WebLogic Server tools.

For information on how to deploy the files using WebLogic Server tools, see Deploy and Undeploy Communication Services and plug-ins.

Managing and Configuring 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	Properties for SOAP2SOAP Plug-ins
Configuration workflow	Configuration Workflow for SOAP2SOAP Plug-ins
List of operations and attributes related to management and provisioning	Provisioning Workflow for SOAP2SOAP Communication Services
Reference of management attributes and operations	Reference: Attributes and Operations for SOAP2SOAP Plug-ins

Properties for SOAP2SOAP Plug-ins

Property	Description
Managed object in Administration Console	<domain name>–>OCSG–><server name>–>Communication Services–><plug-in instance ID>
MBean	Domain=com.bea.wlcp.wlng Name=wlngt InstanceName is same as plug-in instance ID Type=com.bea.wlcp.wlng.httpproxy.management.HTTPProxyMBean
Network protocol plug-in service ID	Defined when generating the SOAP2SOAP plug-in using the Platform Development Studio. When generated fm the Administration console, the ID is <Name>_soap_plugin.
Network protocol plug-in instance ID	The ID is given when the plug-in instance is created, see Managing and Configuring the Plug-in Manager.

Property	Description
Supported Address Scheme	Defined when generating the SOAP2SOAP plug-in using the Platform Development Studio. When generated fm the Administration console, the address scheme is always an empty string.

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 Managing and Configuring the Plug-in Manager for information on how to list the interfaces.

Service type	Given when the plug-in was generated using the Platform Development Studio or the administration console.
Exposes to the service communication layer a Java representation of:	Depends on the WSDLs used.
Interfaces with the network nodes using:	The same protocol as exposed by the application-facing interfaces.
Deployment artifacts	<communication service identifier>_<protocol>_plugin.jar, <communication service identifier>_service.jar and <communication service identifier>_callback.war, packaged in wlng_nt_<communication service>.ear <communication service identifier>_callback.jar and <communication service identifier>.war, packaged in wlng_at_<communication service identifier>.ear <communication service identifier> was given in the Eclipse wizard or the Administration console when the communication service was generated. <protocol> is configurable when using the Eclipse wizard. If the Communication Service was generated using the Administration console, it is always `soap`.

Configuration Workflow for SOAP2SOAP Plug-ins

Below is an outline for configuring the plug-in using the Oracle Communications Services 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 the HTTP time-out behavior:
   - Attribute: HttpTimeoutPeriod
   - Attribute: HttpTimeoutThreshold
   - Attribute: HttpWaitTime
   - Attribute: ReactivateTime
7. Specify heartbeat behavior, see Configuring Heartbeats.
8. 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.

9. 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.

10. 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: HttpTimeoutPeriod

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: HttpTimeoutPeriod

Scope: Cluster

Unit: Seconds

Format: int

Specifies a time period during which a number of HTTP time-outs are counted.

Attribute: HttpTimeoutThreshold

Scope: Cluster

Unit: n/a

Format: int

Specifies the number of HTTP time-outs allowed during a time-period. The time-period is specified in Attribute: HttpTimeoutPeriod.

If the number of time-outs are exceed, the plug-in instance transfers to state disconnected, which means that it does not accept new requests. It stays disconnected during the time-period defined in Attribute: ReactivateTime.

Attribute: HttpWaitTime

Scope: Cluster

Unit: seconds

Format: Int

Specifies the HTTP time-out value. If this value is exceeded, the request is considered to be lost and the counter that tracks the number of requests that encountered HTTP time-outs is increased.

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: ReactivateTime

Scope: Cluster

Unit: seconds

Format: int

Specifies the time a plug-in instance is kept in state disconnected when the number of HTTP time-outs have exceeded the maximum number, specified in Attribute: HttpTimeoutThreshold, over a time-period, specified in Attribute: HttpTimeoutPeriod.

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 33-3 addApplicationEndPoint

addApplicationEndPoint	
Parameter	**Description**
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 33-4 getApplicationEndPoint

getApplicationEndPoint	
Parameter	**Description**
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 33-5 listApplicationEndPoints

listApplicationEndPoints	
Parameter	**Description**
-	-

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 33-6 removeApplicationEndPoint

removeApplicationEndPoint	
Parameter	**Description**
AppinstanceId	The application instance ID associated with the callback URL.

CHAPTER 34

Configuring Heartbeats

The following section describes how to configure heartbeats for HTTP-based stateless network protocol plug-ins.

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.wlng.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 functionary. 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.wlng.heartbeat.management.HeartbeatMBean

Below is a list of attributes for configuration:

- Attribute: ContentMatch
- Attribute: Enabled
- Attribute: Interval
- Attribute: NetworkServiceUrl

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

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.

CHAPTER 35

Managing and configuring Shortcode mappings

The following section describes how to configure shortcode mappings for incoming messages to Oracle Communications Services 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

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 35-1 addShortCodeMapping

addShortCodeMapping	
Parameter	**Description**
orig_dest_pattern	Pattern that is matched against the original destination address as of a message received from the network. The pattern should be specified as a regular expression: ^6
trans_dest_addr	The resulting translated address. Addresses must be specified with the prefix “tel:”.

Operation: listShortCodeMappings

Scope: Cluster

Displays a list of shortcode mappings.

Signature:

```
listShortCodeMappings()
```

Table 35-2 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 35-3 removeShortCodeMapping

removeShortCodeMapping	
Parameter	**Description**
OriginalDestination	Registered original destination pattern.
MappedDestination	Registered translated destination address.

CHAPTER 36

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

Oracle Communications Services Gatekeeper communication services use an internal service, Parlay Access, 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 Oracle Communications Services 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 Oracle Communications Services Gatekeeper since Oracle Communications Services 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 Oracle Communications Services 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 Oracle Communications Services 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 an Oracle Communications Services 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 Oracle Communications Services 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 Oracle Communications Services Gatekeeper administrator must choose one of these connection modes, and use the same mode for all Oracle Communications Services 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 Oracle Communications Services 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 Oracle Communications Services 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 Oracle Communications Services 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 Oracle Communications Services Gatekeeper administrator must generate one, and distribute it to the OSA/Parlay Gateway administrator. The associated key must be stored in the Oracle Communications Services 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 Oracle Communications Services Gatekeeper connects to the OSA/Parlay Gateway.
   a. If Oracle Communications Services Gatekeeper connects to the OSA/Parlay Gateway as one single user, register this user: see Creating an OSA client.
   b. If Oracle Communications Services 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 Oracle Communications Services 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 Oracle Communications Services Gatekeeper 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 Oracle Communications Services Gatekeeper 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 a Certificate Authority, the private key and certificate for the client and store them on the local file system of the Oracle Communications Services Gatekeeper 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 Oracle Communications Services Gatekeeper level.

Note: One mapping must be created for each OSA/Parlay SCS (network service) the Oracle Communications Services 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.wlng.parlay.access.ParlayAccessMBean

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

- 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 36-1 activateMapping

activateMapping	
Parameter	**Description**
id	ID of the OSA/Parlay client mapping to activate. See Operation: listMappings.

Operation: addClient

Scope: Cluster

Adds an OSA/Parlay Client.

Signature:

```
addClient(osaClientAppId: String, clientKeyFile: String, clientCertFile:
String, clientKeyPwd: String, keystorePwd: String)
```

Table 36-2 addClient

addClient	
Parameter	**Description**
osaClientAppId	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: `<Enterprise Operator>\<Application ID>` Example: myEntopId\myAppId
clientKeyFile	The directory path (including file name) to the private key for the OSA Client. **Note:** This path is on the file system of the Oracle Communications Services Gatekeeper Network Tier server. Leave empty if not authenticating using certificates.
clientCertFile	The directory path (including file name) to the certificate for the OSA Client. The certificate is provided in order to verify the private key is correct. **Note:** This path is on the file system of the Oracle Communications Services Gatekeeper Network Tier server. Leave empty if not authenticating using certificates.
clientKeyPwd	The password for the private key. Leave empty if not authenticating using certificates.
keystorePwd	The keystore's password as defined when configuring the Oracle Communications Services Gatekeeper, see Operation: setKeyStorePassword.

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 36-3 addConnection

addConnection	
Parameter	**Description**
gwId	The ID of the OSA/Parlay Gateway, as returned when the OSA Gateway was created. See Operation: addGw. Also see Operation: listGw.
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: `parlay_initial`. Use path syntax to specify recursive naming contexts. Example: `/parlay/fw/parlay_inital` 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 36-4 addConnectionIOR

addConnection	
Parameter	**Description**
gwId	The ID of the OSA/Parlay Gateway, as returned when the OSA Gateway was created. See Operation: addGw. Also see Operation: listGw.
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_inital` 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 36-5 addGw

addGw	
Parameters	**Description**
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 Oracle Communications Services 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 36-6 AddMapping

addMapping	
Parameter	**Description**
serviceProviderID	ID of the service provider account the application is associated with. **Note:** If left empty, the mapping will *not* be applied on service provider account and application account level.
applicationID	ID of the application account. **Note:** If left empty, the mapping will *not* be applied on application account level.
serviceType	OSA/Parlay service type name (`TpServiceTypeName`) of the OSA/Parlay SCS to which the OSA Client is to be mapped. See the specification for the OSA/Parlay Framework for a list of recommended service type names.
osaClientAppId	The OSA/Parlay account's `clientAppID`, a string consisting of the entOpId followed by \, followed by the appId. Example: `sp1\app1`. The entOpId and appId is provided by the OSA Gateway administrator.
properties	OSA/Parlay service properties to be used in the look up (service discovery) phase when requesting a service (OSA/Parlay SCS) from the OSA/Parlay Gateway. The properties are specified as a space separated list in the following way: `<propname1> <propval1> <propname2> <propval2>` The properties varies between OSA/Parlay Gateway implementations.
authType	Authentication type to be used. The type is defined according to the OSA/Parlay standard. P_AUTHENTICATION is the only supported. **Note:** When `P_AUTHENTICATION` is used, no encryption or signing algorithm will be used and the parameters encryptionMethod and signingAlgorithm can be left empty.
encryptionMethod	Method used for encryption. The type is defined according to OSA/Parlay standard. If the type is not specified, enter `P_RSA_1024`.

Table 36-6 AddMapping

addMapping	
Parameter	**Description**
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 Operation: addGw, and Operation: listGw.
initConnection	Indicating if the connection to OSA/Parlay Gateway should be initialized immediately. That is, if authentication should performed when the Operation: addClient operation is invoked.

Operation: listActiveMappings

Scope: Cluster

Lists the IDs for active OSA/Parlay Client Mappings.

Signature:

```
listActiveMappings()
```

Returns a list of IDs for active mappings.

Table 36-7 listActiveMappings

listActiveMappings	
Parameter	**Description**
-	-

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 36-8 listActiveMappingsForGw

listActiveMappingsForGw	
Parameter	**Description**
gwId	The ID of the OSA Gateway.

Operation: listGw

Scope: Cluster

Lists the IDs of all registered OSA/Parlay Gateways.

Signature:

```
listGw()
```

Table 36-9 listGw

listGw	
Parameter	**Description**
-	-

Operation: listMappings

Scope: Cluster

Lists the configured OSA/Parlay Client Mappings.

Signature:

```
listMappings()
```

Table 36-10 listMappings

listMappings	
Parameter	**Description**
-	-

Operation: removeClient

Scope: Cluster

Removes an OSA/Parlay client.

Signature:

```
removeClient(osaClientAppId: String, keystorePwd: String)
```

Table 36-11 removeClient

removeClient	
Parameter	**Description**
osaClientAppId	The OSA/Parlay client application ID (and alias in keystore). See Operation: addClient.
keystorePwd	Oracle Communications Services Gatekeeper keystore password.

Operation: removeConnection

Scope: Cluster

Removes an OSA/Parlay Gateway Connection

Signature:

```
removeConnection(gatewayId: int, connectionId: int)
```

Table 36-12 removeConnection

removeConnection	
Parameter	**Description**
gatewayId	The ID of the OSA/Parlay Gateway.
connectionId	The ID of the connection. The ID was returned when the connection was setup, see: • Operation: addConnection • Operation: listActiveMappings • Operation: listMappings

Operation: removeGw

Scope: Cluster

Removes an OSA/Parlay Gateway.

Signature:

```
removeGw(id: int, keystorePwd: String)
```

Table 36-13 removeGw

removeGw	
Parameter	**Description**
id	The ID of the OSA/Parlay Gateway to remove. The ID was returned when the OSA Gateway was created, see • Operation: addGw • Operation: listGw
`keystorePwd`	The Oracle Communications Services Gatekeeper keystore password.

Operation: removeMapping

Scope: Cluster

Removes an OSA/Parlay client mapping.

Signature:

```
removeMapping(id: int)
```

Parameters:

Table 36-14 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 36-15 `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 36-16 viewActiveMappingState

viewActiveMappingState	
Parameter	**Description**
mappingId	The ID of the OSA Client Mapping.

CHAPTER 37

Resolving Policy Deny Codes

The following section describes policy deny codes generated by the Service Interceptors:

Introduction

Policy deny codes are embedded in Policy Exception thrown to applications, embedded in CDRs and alarms.

Policy exceptions are thrown when a usage policies has been violated by an application, for example if a parameter provided by and application is out of bounds or a that the request rate has been exceeded.

In Policy Exceptions thrown towards applications, the deny code is embedded in the exception.

In alarms, the policy deny code and the is present if the alarm originated from a policy violation.

In CDRs, the policy deny code is present in the additional information field, enclosed by <denyCode>. Example: <denyCode>21</denyCode>.

Policy Deny Data

Table 37-1 contains a list of policy deny codes data generated when a policy deny occurs.

Table 37-1 Policy deny data

Policy Deny Code	ID of corresponding alarm	TagPolicy field in alarm	Description
-1	Not applicable	Not applicable	Unspecified
0	Not applicable	Not applicable	Unspecified
1	102826	1001	Application instance does not exist.
2	102826	1001	Application instance is not active.
3	102826	1001	Application does not exist.
4	102826	1001	Service provider account does not exist.
6	102827	1002	Unable to get service provider and application information.
7	102823/3026	2003	Application request limit exceeded.
8	102823/3026	2003	Service provider request limit exceeded.
9	102828	2001	Application request limit exceeded for Service Type.
10	102828	2001	Service provider request limit exceeded for Service Type.
11	102826	1001	Service provider account is not active.
12	102826	1001	Application instance is not active.
13	102822/3025	2002	Service provider quota limit exceeded.
14	102822/3025	2002	Application quota limit exceeded.

Table 37-1 Policy deny data

Policy Deny Code	ID of corresponding alarm	TagPolicy field in alarm	Description
15	102829	2008	Service provider quota limit exceeded for Service Type.
16	102829	2008	Application quota limit exceeded for Service Type.
20	102830	4001	All properties denied.
21	102831	3001	Parameter value is not allowed.
22	102832	4002	Request info is empty.
23	102833	3002	Accessing the method is not allowed.
24	102834	3003	No service provider group SLA found for application instance.
25	102834	3003	No application group SLA found for application instance.
26	102835	4003	Exception thrown calling correlator.
27	102836	4004	Exception thrown calling factory.
28	102824/3027	2004	Service provider node request limit exceeded.
29	102825/3028	2005	Global or service provider node contract is out of date.
30	102837	3004	No global or service provider node SLA found.
31	102838	3005	Application or Service provider group service contract is out of date.
32	102839	3006	Application or service provider Service Type contract is out of date.

Table 37-1 Policy deny data

Policy Deny Code	ID of corresponding alarm	TagPolicy field in alarm	Description
33	102834	3003	No SLA found.
34	102840	3007	No Service Contract found.
35	102841	2006	Subscriber restrict all.
36	102842	2007	Subscriber quota limit reached.
37	102842	2007	Subscriber rate limit reached.
38	102824	2004	Global node request limit exceeded.
104	n/a	n/a	The request has been denied through the credit control interceptor