

Oracle® Communications Services Gatekeeper

Platform Test Environment User's Guide

Release 6.0

E50765-01

January 2015

Copyright © 2008, 2015, Oracle and/or its affiliates. All rights reserved.

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

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

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

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface	v
Audience	v
Downloading Oracle Communications Documentation	v
Documentation Accessibility	v
Related Documents	v
1 Understanding the Platform Test Environment	
Introducing the PTE	1-1
Introducing the PTE Graphical User Interface	1-4
PTE Configuration Workflow	1-5
Extending the PTE	1-6
2 Using the PTE to Test Services Gatekeeper	
Starting the PTE	2-1
Configuring and Maintaining Your PTE Server Environment	2-2
Connecting the PTE to Services Gatekeeper	2-2
Configuring Communication Services by Changing MBean Attributes and Operations	2-3
Creating a Script to Capture Your MBean Changes	2-4
Running MBean Scripts	2-4
Tracking Communication Service Behavior	2-5
Tracking Server Behavior with the Server Log	2-6
Configuring and Managing a Database Connection	2-7
Viewing Database Tables and Fields	2-7
Running SQL Commands on the Database	2-8
Configuring Application-Facing Clients to Test	2-9
Configuring Simulated Telecom Networks	2-10
Configuring Network Elements on the Maps Panel	2-10
Changing the Map Image	2-12
Changing the Map Range	2-12
Configuring Network Protocols to Test With	2-13
Configuring Diameter Billing Tests	2-13
Configuring a Web Server for HTTP Testing	2-15
Configuring and Testing Service-Level Agreements	2-17
Configuring and Testing Composed Service-Level Agreements	2-20
Starting the Composed SLA Editor	2-20

Downloading an Existing Composed SLA	2-21
Creating a New Composed SLA.....	2-22
Editing an SLA	2-22
Upload an Edited Composed SLA	2-26
Monitoring Budget Use	2-26

3 Adding and Testing Custom Client Modules

Adding Custom PTE Client Modules to the PTE.....	3-1
Testing New or Changed Client Modules.....	3-1
Understanding Communication Service Unit Tests	3-1
Understanding the Example Unit Test	3-2
Creating and Running a New Unit Test	3-5

Preface

This document describes the Platform Test Environment (PTE), which is a part of Oracle Communications Services Gatekeeper. The PTE simulates an end-to-end telecom network, including Services Gatekeeper, applications, mobile terminals, and network elements. You use this simulated telecom network to test both default Services Gatekeeper features and your own custom communication services.

Audience

This document is intended for Services Gatekeeper developers who will test default Services Gatekeeper features, and any new communication services that they create using the Platform Development Studio.

Downloading Oracle Communications Documentation

Services Gatekeeper documentation is on the Oracle Technology Network website at <http://docs.oracle.com>.

Additional Oracle Communications documentation is available from the Oracle software delivery website at <https://edelivery.oracle.com>.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Related Documents

For related information, see the following Services Gatekeeper documents:

- *Oracle Communications Services Gatekeeper Concepts*
- *Oracle Communications Services Gatekeeper Multi-tier Installation Guide*
- *Oracle Communications Services Gatekeeper Extension Developer's Guide*

Understanding the Platform Test Environment

Oracle Communications Services Gatekeeper includes the Platform Test Environment (PTE), a graphical user interface (GUI) that you use to test default Services Gatekeeper features and your own custom communication services. This chapter introduces the PTE and provides a configuration workflow for using it.

Introducing the PTE

The PTE simulates both the application-facing interfaces and network-facing protocols so you can test Services Gatekeeper behavior. [Figure 1-1](#) shows an example of the PTE graphical user interface. The **Clients** tool is selected, configuring the tool to test application protocols. The **Messaging** set of application protocols is also selected, and you see tabs for each of the supported messaging protocols available to test. The lower half of the GUI shows the **Map** option for testing a telecom network is selected, and the map is populated with a sample coverage area circle, cell phone, and truck.

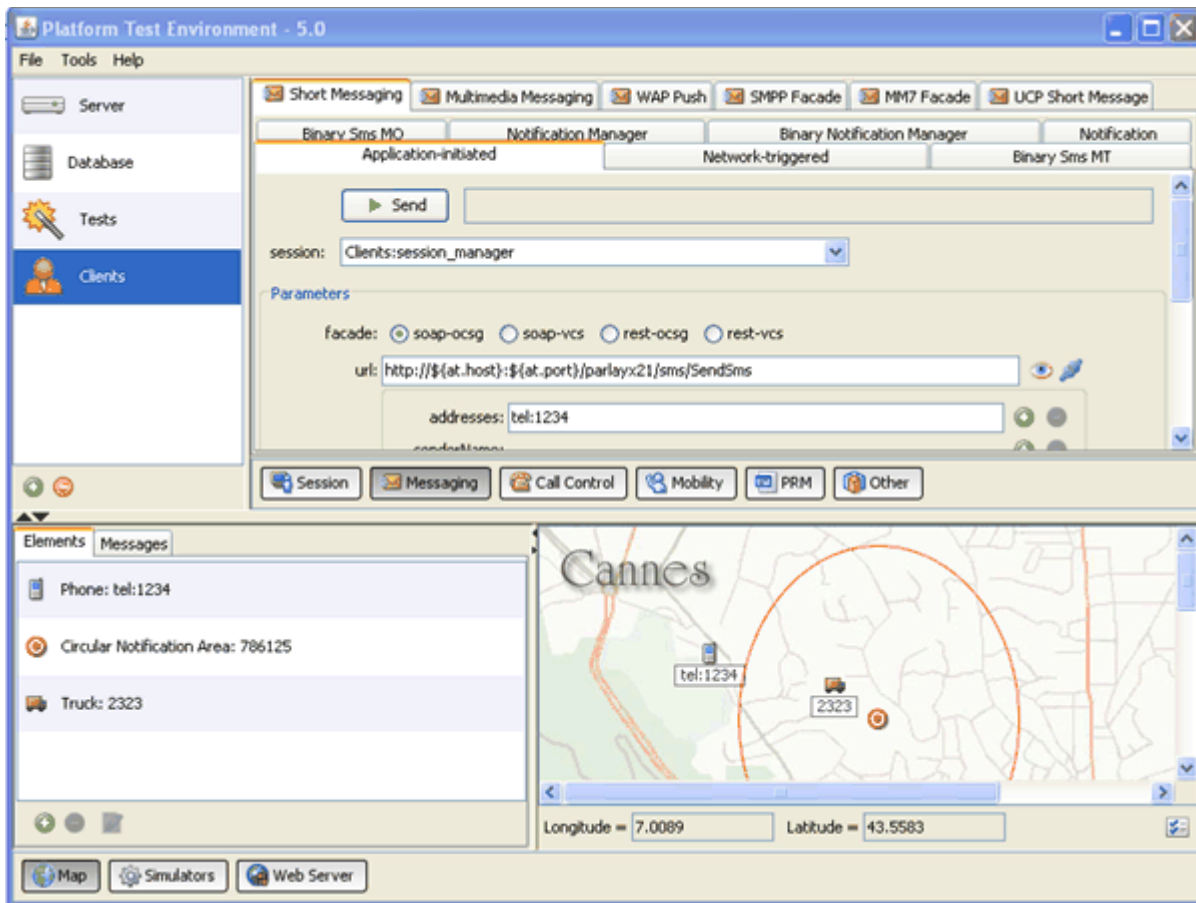
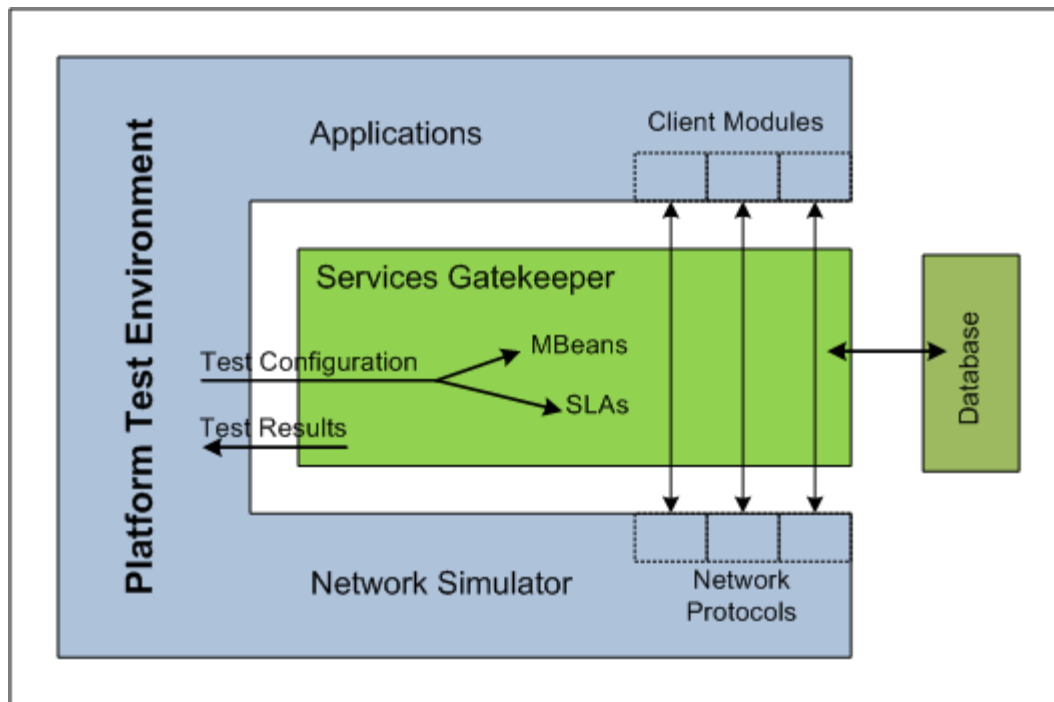
Figure 1–1 The PTE Graphical User Interface

Figure 1–2 shows how the PTE is integrated with Services Gatekeeper. The PTE includes clients, simulators, tools, and plug-ins to mimic most of the default Services Gatekeeper features. The PTE is a complete telecom testing environment. You create actual applications (client module) within the PTE to use as application-facing interfaces. Each client module interacts with a simulated network protocol on the network-facing part of the PTE. The client modules interact with simulated network protocols just as they would in a production environment.

You use the PTE to configure an actual running Services Gatekeeper implementation using MBeans and specify access and budgets with service-level agreements (SLAs) just as you would in a production environment. Test results are returned to the PTE during testing.

You can also extend the PTE to test your own custom features. After testing is complete, you can connect the PTE GUI to your production Services Gatekeeper installation and either continue testing or download your tested changes to the production system. You configure SLAs using a graphical SLA Editor and use the real-time graphic Budget Monitor to see how your changes affect the rates and quotas. A Diameter-based simulator allows you to test billing.

Figure 1-2 PTE/Services Gatekeeper Integration



The PTE offers these additional features:

- A default set of client modules for most default communication services that use both the web services facade and the RESTful facade.
- Partner Relationship Management (PRM) test clients for many operations covered by the PRM interfaces.
- Simulators for most network protocols supported by the default communication services.
- A log browser for checking server logs.
- A Java Message Service-based Event Data Record (EDR)/Customer Data Record (CDR)/alarm listener.
- A database browser for interacting with the database.
- Real-time duration test graphing.
- An embedded TCP monitor.
- An easily extendable architecture, including:
 - An example application test client for use with the example communication service.
 - An example network simulator for use with the example communication service.
 - A set of Service Provider Interfaces (SPIs) that allow your modules to interact with the PTE.
- A framework for building unit tests, including:
 - A base test class, derived from JUnit.
 - Mechanisms that simplify connecting to the PTE.

- An example test case for use with the example communication service.

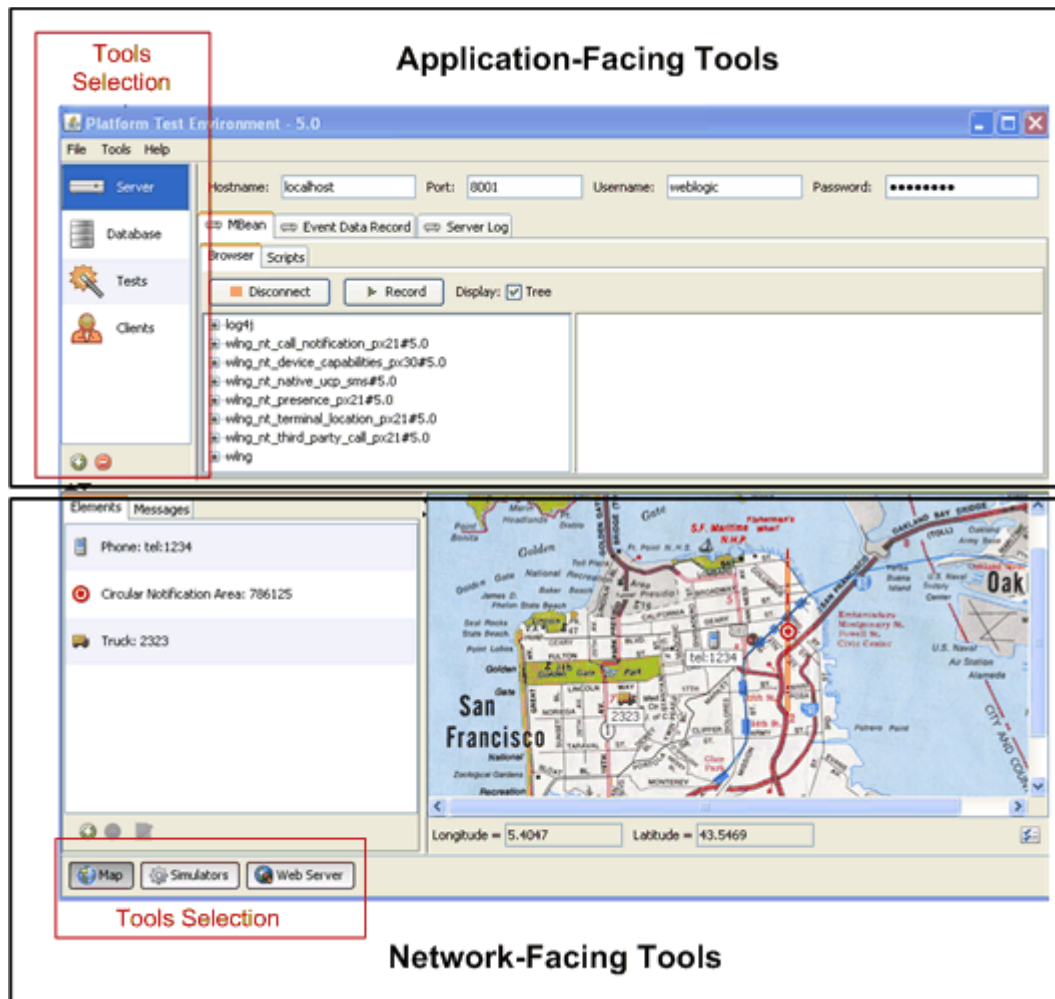
Introducing the PTE Graphical User Interface

The main PTE GUI components are:

- An application-facing configuration panel (shown in [Figure 1-3](#)) in the upper right of the GUI. You use this to select a server to connect to, a database to log on to, and individual applications (clients) to configure.
- The application-facing configuration panel includes a tools selection panel (shown in [Figure 1-3](#)) in the upper left of the GUI that you use to select a PTE tool to configure or test.
- A network-facing simulation panel (shown in [Figure 1-3](#)) in the lower part of the GUI that you use to simulate telecom network settings so you can test client and network behavior.
- The network-facing simulation panel also contains a tools selection panel (shown in [Figure 1-3](#)) in the lower left of the PTE GUI that you use to select a network simulation tool to configure.
- An SLA editor that you use to configure and test SLAs. The SLA editor is a graphical editor that you use to edit SLA files.
- The SLA editor also contains a contract editor panel that you use to view SLA rates, restrictions, and quotas that individual applications are allowed.
- A Budget Monitor that shows you the effect that testing services has on existing Services Gatekeeper rates and quotas.

[Figure 1-3](#) shows the main panels of the PTE GUI, with the application-facing tools area on top, and the network-facing tools on the bottom. The tools selection panels for both panels are outlined for emphasis.

Figure 1–3 The Main PTE GUI Sections



Many PTE windows offer icon shortcuts for common tasks. Hover over these icons, and tooltips appear.

To run the PTE, you must have set up application and service provider accounts in Services Gatekeeper, associated them with appropriate groups, and attached SLAs. You must have also configured the communications services that you are testing and created instances for those communication services as required. For a detailed description of these and other Services Gatekeeper management tasks, see:

- *Services Gatekeeper System Administrator's Guide*
- *Services Gatekeeper Communication Service Reference Guide*
- *Services Gatekeeper Portal Developer's Guide*

PTE Configuration Workflow

Using the PTE to test any of the default Services Gatekeeper features involves the following steps:

1. Use the Plug-in Manager to create a plug-in instance for each communication service to test. For information, see "Configuring and Managing the Plug-in Manager" in *Services Gatekeeper System Administrator's Guide*.

2. Use the Plug-in Manager to set up routing for the plug-in instances. For information, see "Configuring and Managing the Plug-in Manager" in *Services Gatekeeper System Administrator's Guide*.
3. Install and start a database for Services Gatekeeper to use. For information, see "Installing the Database" in *Services Gatekeeper Multi-tier Installation Guide*.
4. Install and start Services Gatekeeper. For information, see "Installing Services Gatekeeper" in *Services Gatekeeper Multi-tier Installation Guide*.
5. Install and start the PTE. See "[Starting the PTE](#)" for information.
6. Connect your running PTE to your Services Gatekeeper implementation (database). For information, see "[Connecting the PTE to Services Gatekeeper](#)".
7. Using the PTE, configure your application-facing applications (clients). For information, see "[Configuring Application-Facing Clients to Test](#)".
8. Using the PTE SLA Manager, create an application account, service provider accounts, service provider groups, application groups. For information, see "[Configuring and Testing Service-Level Agreements](#)".
9. Tie your new SLA components into the PTE. For information, see *Services Gatekeeper Portal Developer's Guide*.
10. Using the Plug-in Manager, set up SLA routing. For information, see "Configuring and Managing the Plug-in Manager" in *Services Gatekeeper System Administrator's Guide*.
11. Configure the applications (clients) using the PTE application-facing tools) For information, see "[Configuring Application-Facing Clients to Test](#)".
12. Configure the PTE network simulator. For information, see "[Configuring Simulated Telecom Networks](#)".

Extending the PTE

The default PTE implementation is preconfigured to test the default Services Gatekeeper communication services. You can customize these services or create your own new services. For information, see "Extending the ATE and the PTE" in *Services Gatekeeper Extension Developer's Guide*.

After you have created or customized communication services, see "[Adding and Testing Custom Client Modules](#)" for testing instructions and a sample unit test.

Using the PTE to Test Services Gatekeeper

This chapter explains how to use the Platform Test Environment (PTE) graphical user interface to test an Oracle Communications Services Gatekeeper implementation. This GUI is the main access point for most PTE features. Any changes you make using the PTE are saved when you exit the program.

This chapter assumes that you have read "[Understanding the Platform Test Environment](#)" and understand the PTE configuration workflow.

Starting the PTE

Before you use the PTE, you must first:

- Install Services Gatekeeper PTE. For installation instructions, see "Installing the Platform Test Environment" in *Services Gatekeeper Multi-tier Installation Guide*.
- Run the `Middleware_home/wlserver/server/bin/setWLSEnv.sh` script or set the equivalent path so that you have access to the PTE scripts in `Middleware_home/ocsg_pte`.

To start the PTE on a Windows-based system, do one of the following:

- (GUI Mode) From your system **Start** menu, select **All Programs**, then **Oracle Communications Services Gatekeeper 6.0.0.0**, and then **Platform Test Environment**.
- (GUI Mode) Using a command window, go to `Middleware_home/ocsg_pte` and enter the following:

```
run.cmd
```
- (Console Mode) Using a command window, go to `Middleware_home/ocsg_pte` and enter the following. Console mode is usually used for testing new or customized PTE extensions.

```
./console.cmd
```

To start the PTE on a Solaris or Linux-based system, do one of the following:

- (GUI Mode) Using a command window, go to `Middleware_home/ocsg_pte` and enter the following:

```
./run.sh
```
- (Console Mode) Using a command window, go to `Middleware_home/ocsg_pte` and enter the following. Console mode is usually only used for testing new or customized PTE extensions.

```
./console.sh
```

Note: Settings compatibility between different versions of the PTE is *not* guaranteed.

Configuring and Maintaining Your PTE Server Environment

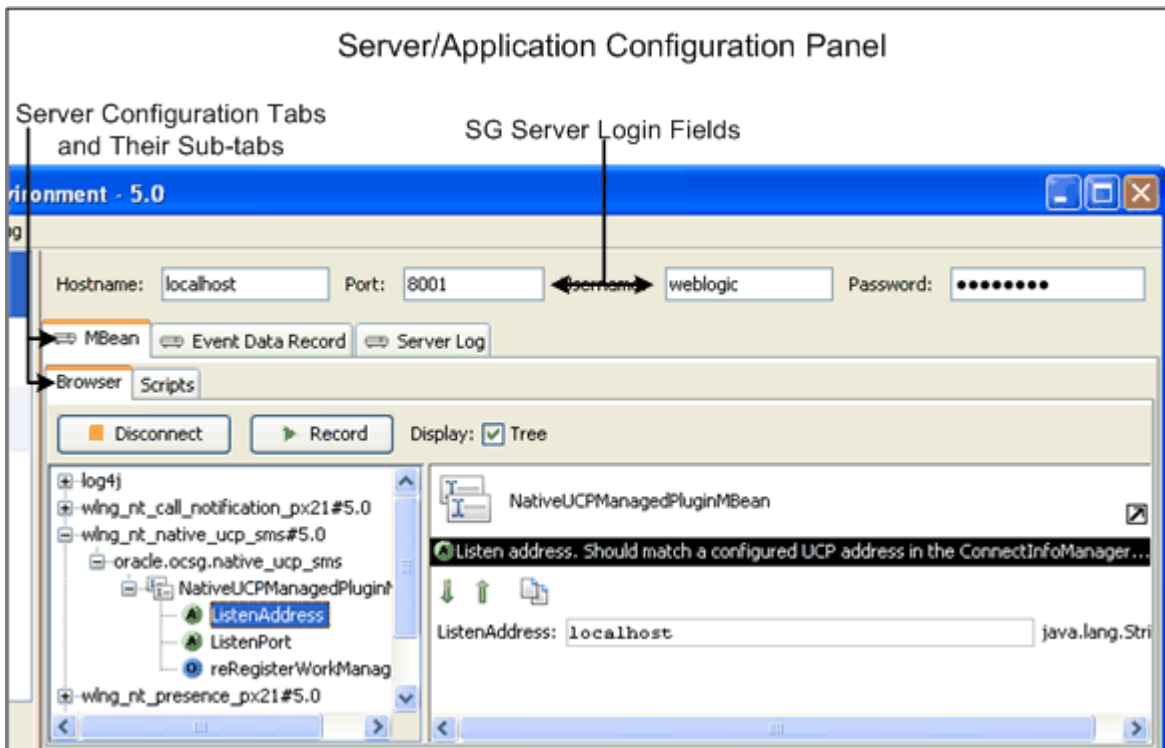
To configure and maintain your PTE server environment, you perform these tasks using the PTE **Server** panel:

- [Connecting the PTE to Services Gatekeeper](#)
- [Configuring Communication Services by Changing MBean Attributes and Operations](#)
- [Creating a Script to Capture Your MBean Changes](#)
- [Tracking Communication Service Behavior](#)
- [Tracking Server Behavior with the Server Log](#)

Connecting the PTE to Services Gatekeeper

The PTE requires a connection to a running Services Gatekeeper implementation through a database domain. You can use PTE to control any running Services Gatekeeper server by changing the **Hostname**, **Port** number, and an administrative **Username** and **Password**. [Figure 2-1](#) shows where you enter Services Gatekeeper login fields. The Server Configuration tabs and their sub-tabs are also marked.

Figure 2-1 The Server/Application Configuration Panel

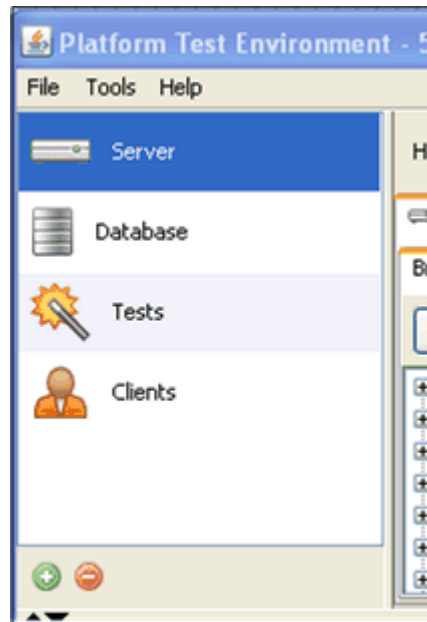


To connect to Services Gatekeeper:

1. Do one of the following:
 - If you started the PTE from your system's **Start** menu, it automatically connects to the corresponding Services Gatekeeper implementation.
 - To connect to a different Services Gatekeeper implementation, enter the **Hostname, Port, Username** and **Password** that you configured in that database domain and click **Connect**.
2. In the tools selection panel, click **Server**.

Figure 2–2 shows the application-facing tools selection panel with the Server item selected.

Figure 2–2 The Application-facing Tools Selection Panel



3. Click the **MBean** sub-tab.
4. Click **Connect**. The MBean tree is populated.

To switch to a different Services Gatekeeper implementation:

1. Click **Disconnect**.
2. Enter the **Hostname, Port, Username**, and **Password** that you configured in the database domain.
3. Click **Connect**.

Configuring Communication Services by Changing MBean Attributes and Operations

You configure communication services by changing their MBean attributes and operations. You use the **MBean** tab to change individual MBean attributes and perform MBean operations. These changes persist in the database. You can also record a series of changes in a script and use the script to repeat the changes as needed.

This section assumes that you know the configuration changes you want to make for each communication service. For information on the configuration choices, read through the *Services Gatekeeper Communication Service Reference Guide*, a separate document in this set.

To configure a communication service:

1. Connect to a Services Gatekeeper implementation.
2. In the tools selection panel, click **Server**.
3. Click the **MBean** sub-tab.
4. Click the **Browser** sub-tab which displays the list of available MBeans.
5. Select an MBean to modify. The MBean attributes are displayed in the panel to the right of the MBean.
6. Make any changes to the MBean attribute or operation.
7. Click the green down arrow (for attributes) or right-facing triangle (for operations) to save your changes to the database.
8. Select and edit any other MBeans as necessary to configure the communication service.

Creating a Script to Capture Your MBean Changes

You can record and repeat your MBean configuration changes using the **Scripts** sub-tab. You can run these scripts manually or programmatically later as needed.

To create an MBean scrip to capture MBean changes:

1. Connect to a Services Gatekeeper implementation.
2. In the tools selection panel, click **Server**.
3. Click the **MBean** sub-tab.
4. Click **Record**.

The Recording starts.

5. Perform the MBean tasks to capture.

Click **Recording**.

6. Enter a name for the script and click **OK**.

Your new script appears in the **Scripts** sub-tab.

Running MBean Scripts

MBean scripts automate the process of changing your MBean operations and attributes. The instructions in this section run scripts within a single PTE implementation.

To run an MBean script:

1. Connect to a Services Gatekeeper implementation.
2. In the tools selection panel, click **Server**.
3. Click the **Scripts** sub-tab.
4. Select a scrip to run.
5. Click **Run**.

Tracking Communication Service Behavior

Services Gatekeeper keeps track of communication service behavior by recording Event Data Records (EDRs) at logical points as the communication service communicates with applications, the network, and the database. You can track and debug communication service behavior by viewing the EDRs in the PTE. For information on the list of EDRs generated by each communication service, see *Services Gatekeeper Communication Service Reference Guide*.

You use the **Event Data Record** tab to view EDR data. [Figure 2-3](#) shows the Event Data Record tab populated with EDRs. EDR data is shown, including the EDR number, Type, ID number, a timestamp, and the EDR description. The Dynamic Filter boxes that limit the EDRs displayed are marked, as is the Counter Mode check box. The EDR key/value pairs are shown for EDR number 24 in the detail window at the bottom.

Figure 2-3 The Event Data Record Tab

The screenshot displays the 'EDRs and EDR Detail' window. At the top, there are fields for Hostname (localhost), Port (8001), Username (weblogic), and Password (masked). Below these are tabs for MBean, Event Data Record (selected), and Server Log. A 'Dynamic Filter Check--boxes' section includes 'Disconnect' (checked) and 'Counter Mode' (circled). To the right, 'Show:' filters are checked for 'EDR (12)', 'CDR', and 'Alarm (12)'. The main table lists EDRs with columns for #, Type, ID, Time, and Description. EDR 24 is highlighted. Below the table is a 'Key Value' section for EDR 24, showing details like Position: after, ServerName: AdminServer, ServiceName: TerminalLocation, ServiceProviderId: TermStatusSP, SessionId: app:-680297884505475980, Source: method, TagEdr: 9001, and Timestamp: 1277922641390.

#	Type	ID	Time	Description
19	Alarm	199900	2010-06-30 11:30:20	default exception
20	Edr	9001	2010-06-30 11:30:20	GetLocation
21	Edr	9001	2010-06-30 11:30:41	GetLocation
22	Alarm	111206	2010-06-30 11:30:41	Failed to send location request.
23	Alarm	199900	2010-06-30 11:30:41	default exception
24	Edr	9001	2010-06-30 11:30:41	GetLocation

Key	Value
Position:	after
ServerName:	AdminServer
ServiceName:	TerminalLocation
ServiceProviderId:	TermStatusSP
SessionId:	app:-680297884505475980
Source:	method
TagEdr:	9001
Timestamp:	1277922641390

To view EDRs for a communication service:

1. Connect to a Services Gatekeeper implementation.
2. Run a communication service.
Each communication service generates a unique list of EDRs.
3. In the tools selection panel, click **Server**.
4. Click the **Event Data Record** sub-tab.

5. Click **Connect**.

The list of EDRs generated by the communication service is displayed

6. Select an EDR to view its details in the panel below it.

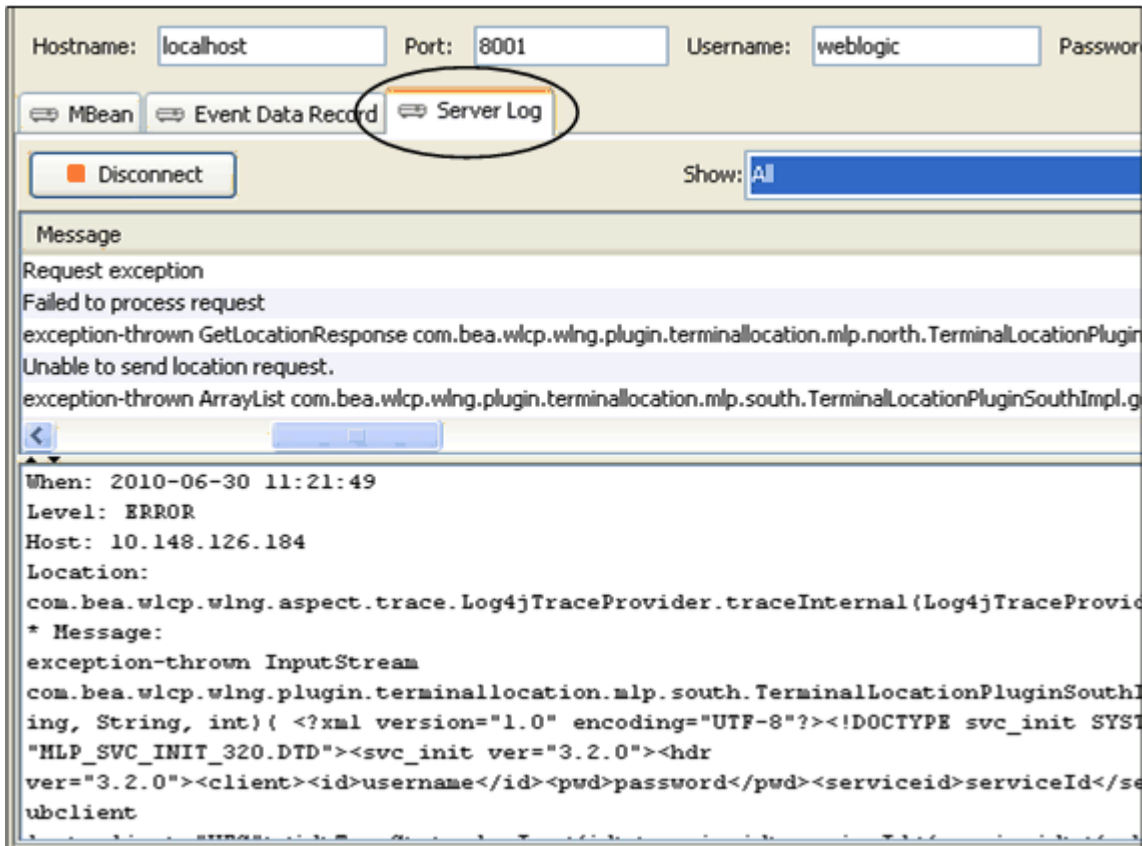
To export, import, or copy to the clipboard a list of EDRs, use the icons in the bottom left of the EDR list window.

To avoid performance degradation in situations where you are expecting a large number of EDRs, check the **Counter Mode** box. This returns a count of the number of records, but does not display the entire contents.

Tracking Server Behavior with the Server Log

Click **Server Log** to view the server logs. [Figure 2-4](#) shows the **Server Log** tab populated with sample server log messages.

Figure 2-4 Server Log Message and Detail Panels



To view EDRs for a communication service:

1. Connect to a Services Gatekeeper implementation.
2. Run a communication service.

Each communication service generates a unique list of EDRs.
3. In the tools selection panel, click **Server**.
4. Click the **Server Log** sub-tab.
5. Click **Connect**.

Server log messages is displayed

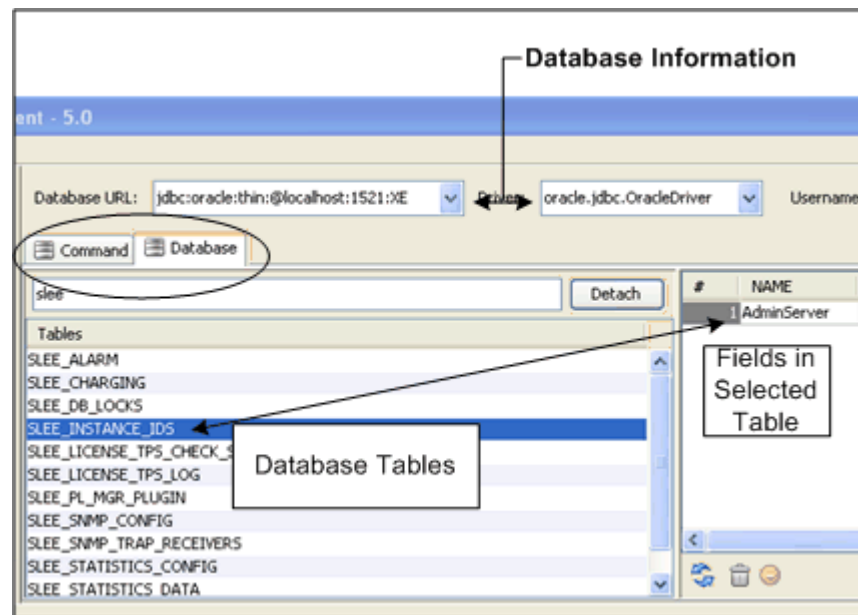
6. Select a log message to view its details in the panel below it.

To export server log messages to a file, click the export icon on the right side of the panel. To clear the list of server logs, click the trash can icon on the right side of the panel.

Configuring and Managing a Database Connection

You can do a lot of testing without connecting the PTE to a database. For example you can verify communication service features and make changes to the SLAs. However, to confirm that your changes make the correct database changes, you need to connect the PTE to a running database. Use the Database Configuration panel to view and change the database connection and database tables. [Figure 2-5](#) shows the Database Configuration panel populated with database tables. A table is selected and its fields shown.

Figure 2-5 The Database Configuration Panel



The Database panel has two sub-tabs:

- Use the **Command** sub-tab to enter SQL commands directly into the database.
- Use the **Database** sub-tab to view database tables and the data in them. To monitor a specific table more closely, select it and click **Detach**. That table appears individually as an additional tab.

Viewing Database Tables and Fields

This procedure requires that you know the database URL, driver, username, and password.

To view database tables and fields:

1. Connect to a Services Gatekeeper implementation.
2. From the tools selection panel, click **Database**.

3. In the **URL** field, enter the URL of a database to connect to.
4. In the **Driver** field, enter the driver of a database to connect to.
5. In the **Username** field, enter the name of a database administrative user in the database to connect to.
6. In the **Password** field, enter the password for the administrative user.
7. Click **Connect**.
8. Click the **Database** sub-tab.

The **Database** sub-tab is populated with database fields. The **Tables** panel on the left displays the database tables.

9. Select a table to display its fields in the panel on the right.

Running SQL Commands on the Database

The PTE database command line allows you to run SQL commands directly on your database. [Figure 2–6](#) shows the **Command** tab window with the SQL command line marked.

This procedure requires a strong knowledge of SQL.

Figure 2–6 The Database Command Tab



This procedure requires that you know the database URL, driver, username, and password.

To run an SQL command against the database:

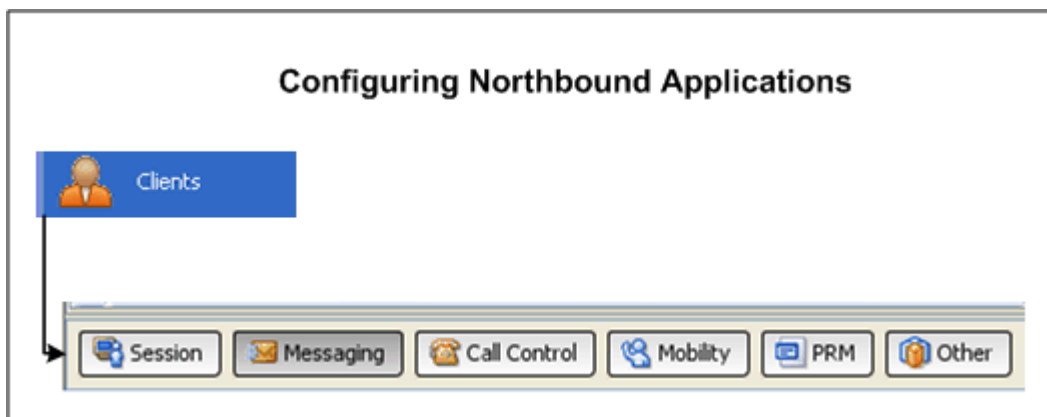
1. Connect to a Services Gatekeeper implementation.
2. From the tools selection panel, click **Database**.
3. In the **URL** field, enter the URL of a database to connect to.
4. In the **Driver** field, enter the driver of a database to connect to.
5. In the **Username** field, enter the name of a database administrative user in the database to connect to.
6. In the **Password** field, enter the password for the administrative user.
7. Click **Connect**.

8. Click the **Command** sub-tab.
The **Command** dialog box is displayed.
9. Enter an SQL command line to execute on the database.

Configuring Application-Facing Clients to Test

Use the **Clients** tool to select and configure your application-facing interfaces (clients) and your Services Gatekeeper connection. [Figure 2–7](#) shows the categories of clients available to configure for testing.

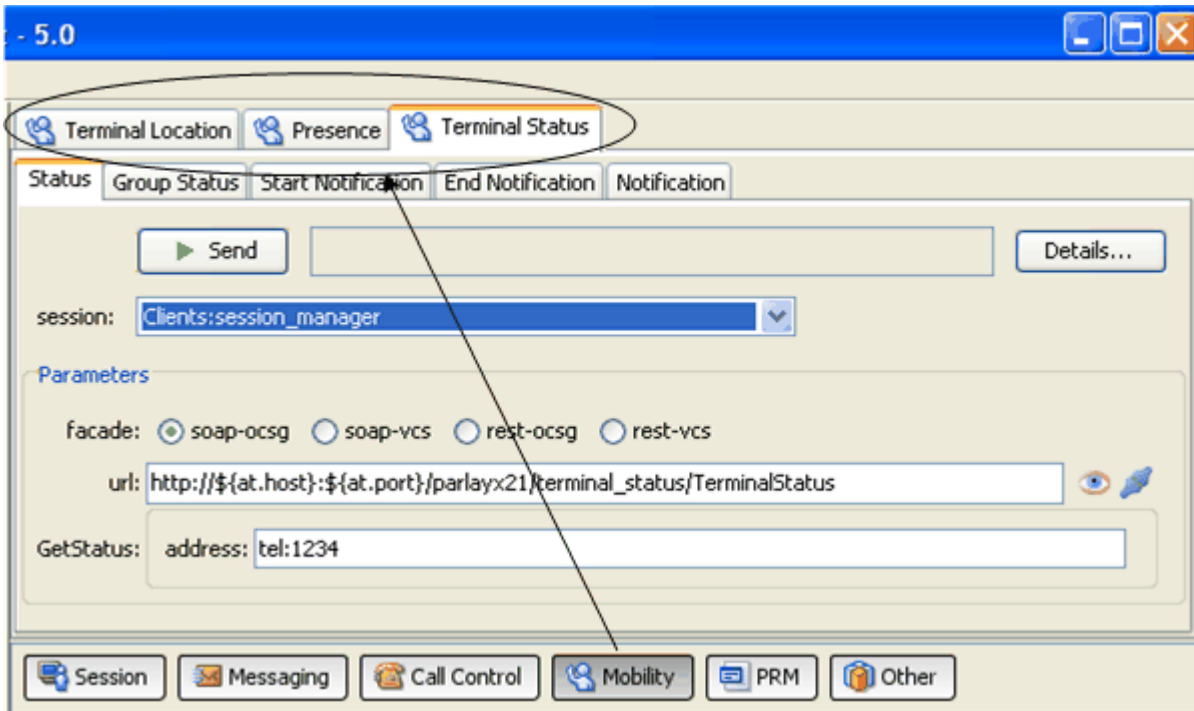
Figure 2–7 *Configuring Application-facing Clients to Test*



- Use the **Session** button to configure the Services Gatekeeper connection.
- Use the **Messaging** button to configure the Service Gatekeeper messaging applications, such as Short Messaging, Multimedia Messaging, and so forth.
- Use the **Call Control** button to configure Audio Call, Call Notification, and Third Party Call applications.
- Use the **Mobility** button to configure Terminal Location, Terminal Status, and Presence applications.
- Use the **PRM** button to configure Partner Relationship Management operator and service provider applications.
- Use the **Other** button to configure Subscriber Profile, Payment, Device Capabilities, and Netex applications.

Each application has unique configuration parameters organized by sub-tab. [Figure 2–8](#) shows the **Mobility** applications, (Terminal Location, Presence, and Terminal Status) available to configure and test. The **Terminal Status** tab is selected, showing the parameters in the **Status** sub-tab.

Figure 2–8 The Mobility Applications



Configuring Simulated Telecom Networks

Each communication service requires a simulated network to test with the PTE. You use the network-facing simulator panel, in the lower part of the PTE, to configure simulated telecom network components and settings. There are three categories to choose from:

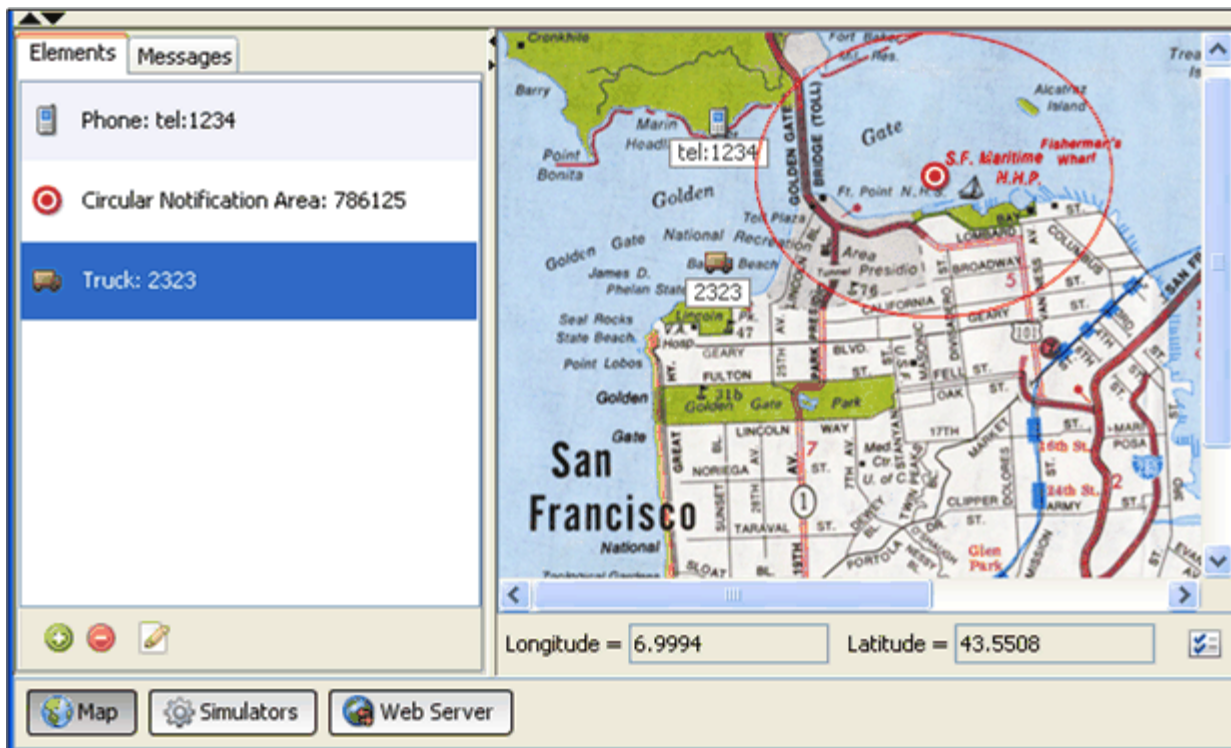
- **Map** displays a test map of a physical location that you can add network components to.
- **Simulator** offers the network protocols available to use as network-facing interfaces. This button also offers the Diameter billing feature.
- **Web Server** simulates an HTTP web server.

Configuring Network Elements on the Maps Panel

The **Maps** panel provides a map on which you can place sample phone terminals, mobile trucks, and connect them with sample coverage contact areas. This offers visual support for testing Parlay X 2.1 SMS, MMS, and Terminal Location traffic and their RESTful equivalents. Use the plus button on the bottom left to create your own sample test elements.

Figure 2–9 shows the **Map** panel with the Maps elements on the left side and the simulated map on the right. The map has an example telephone, truck, and circular notification area deployed on it.

Figure 2–9 The Maps Panel



To add an element to the map:

1. Connect to a Services Gatekeeper implementation.
2. Select the **Map** button at the bottom of the PTE screen.
The map appears.
3. Click the green plus icon at the bottom of the **Elements** sub-tab.
The **Add Element** screen appears.
4. Select the type of element to add.
5. The **Properties** screen appears. Do one of the following:

For a cell phone, enter:

- An **address**.
- A **longitude**.
- A **latitude**.
- An **altitude**.
- A **status** (reachable, unreachable, busy)

For a circular notification area, enter:

- An **id**.
- An **address**.
- A **trigger**.
- A **longitude**.
- A **latitude**.

- A **radius**.
- An **interval**.
- A **duration**.
- A **count**.

For a TS (terminal Status) notification area, enter:

- An **id**
- An **address**.

For a truck, enter:

- An **address**.
- A **longitude**.
- A **latitude**.
- An **altitude**.

6. Click **OK**.

The new element appears on the map.

Changing the Map Image

You can use any GIF or JPEG file for the map image.

To change the map image:

1. Connect to a Services Gatekeeper implementation.
2. Select the **Map** button at the bottom of the PTE screen.
The Map panel appears.
3. Click the Map Properties icon on the bottom right of the map panel.
The Map Properties panel appears.
4. Click the **Custom Image** radio button.
5. Click the **Browse** button and navigate to the location of an image to use for the map.
6. Click **OK**.

The **Map Properties** panel disappears and the new image is displayed in the PTE Map panel.

Changing the Map Range

You can use any longitude and latitude ranges for the map image.

To change the map longitude and latitude ranges:

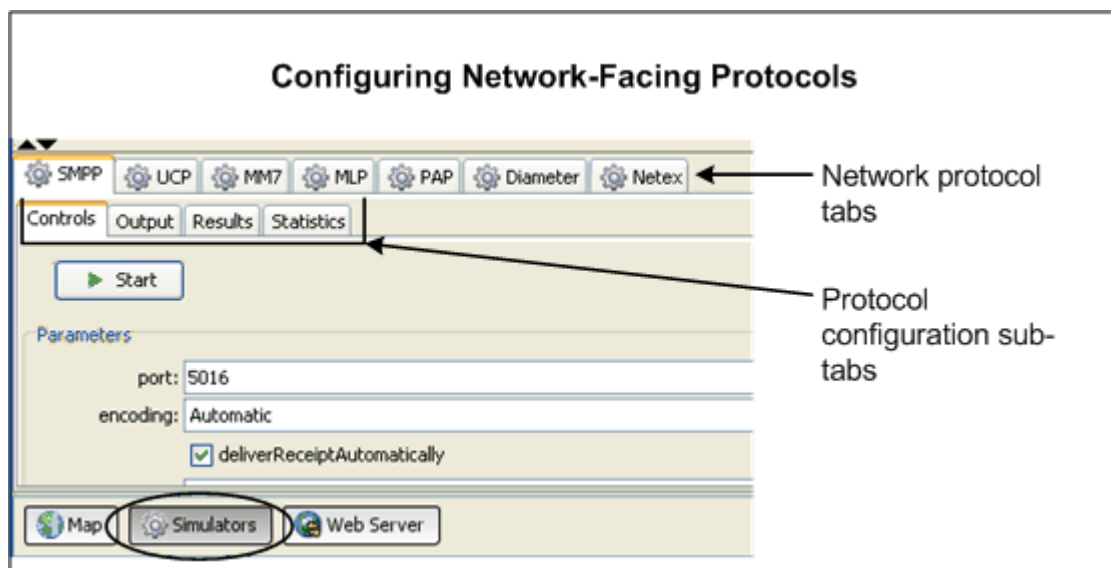
1. Connect to a Services Gatekeeper implementation.
2. Select the **Map** button at the bottom of the PTE screen.
The map appears.
3. Click the Map Properties icon on the bottom right of the map panel.
The Map Properties panel appears.

4. Enter new values for the longitude and latitude ranges.
No validation checking is performed on longitude or latitude. You can enter any values regardless of whether they make sense.
5. Click **OK**.
6. The **Map Properties** panel disappears and the new ranges are displayed in the PTE Map panel.

Configuring Network Protocols to Test With

Each communication service requires a network-facing protocol to test against, and you configure them using the **Simulators** button in the lower part of the PTE. [Figure 2–10](#) shows the network simulator components.

Figure 2–10 Configuring a Network Protocol to Test With



To configure a network protocol for testing:

1. Connect to a Services Gatekeeper implementation.
2. Select the **Simulators** button at the bottom of the PTE.
3. Select a network protocol from the tabs available.
4. Configure, start, and view results from the protocol using the sub-tabs available.

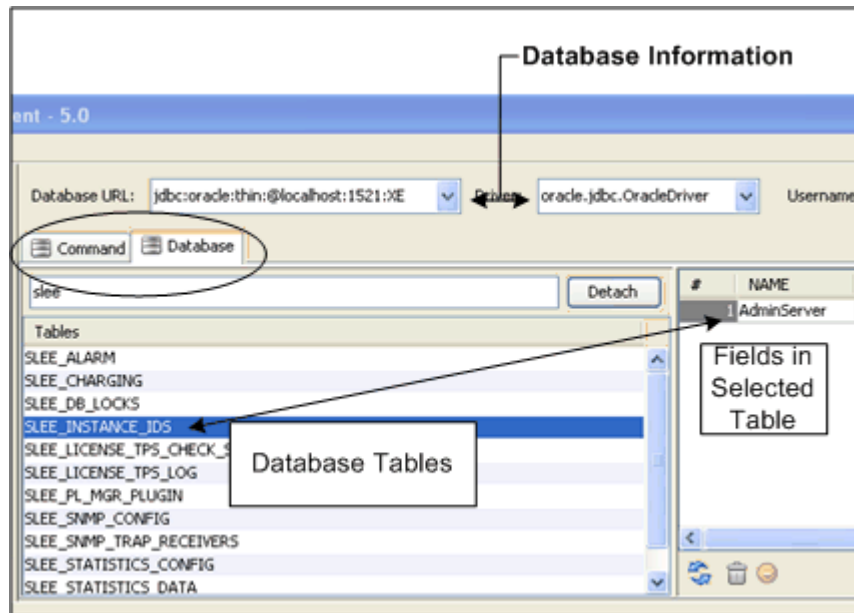
The number and type of protocol configuration sub-tabs offered depends on the individual protocol. In all cases, there is a **Controls** tab to configure the protocol. This area is also where the **Start** button is for each of the modules. The other tabs vary; for example, they may allow you to see the actual content of a message or show you the statistics associated with traffic.

Configuring Diameter Billing Tests

The **Diameter** tab allows you to set up and run any billing simulator integrated with the Diameter protocol. For example, to use the billing simulator with the Payment communication service, create a user in the billing simulator corresponding to the **endUserIdentifier** field in the Payment client, and send a request from the client through the Diameter simulator to the Billing simulator.

Do not use an existing localhost value for the Diameter Origin Host parameter in the Payment communication service. You must use the actual name of a host. [Figure 2–11](#) shows the Diameter simulator **Controls** tab.

Figure 2–11 The Diameter Simulator Controls Tab



To configure Diameter testing:

1. Create a Diameter plug-in instance.
2. Connect the PTE to a Services Gatekeeper implementation.
3. Click the **Simulators** button at the bottom of the PTE GUI.
4. Click the **Diameter** tab.
5. Click the **Controls** sub-tab.
6. Enter a Diameter realm to use.
7. Enter the first Diameter request listener port in the **port1** field. This port can accept either Rf (ACR; post-paid) or Ro (CCR; pre-paid) requests, and can be a multi-home port. The default value is 3588.
8. Enter the second Diameter request listener port in the **port2** field. This port can also accept either Rf (ACR; post-paid) or Ro (CCR; pre-paid) requests, and can be a multi-home port. The default value is 3589.
9. (As needed) Change the **failingSubscriberIdData** identifier.
The default identifier (**123123**) is an Oracle-supplied subscriber account that does not have any funds. This is useful for testing “insufficient funds” scenarios. Change this if you need to.
10. (As needed) Check the **recordBillingEvents** checkbox.
Checking this box directs the PTE to record and display billing events.
11. (As needed) Check the **dumpMessage** checkbox to delete Diameter messages.
Checking this box directs the PTE to delete Diameter messages instead of sending them to the PTE output panel.

12. (As needed) Enter an alternate value for **maxRecordBillingEvents**.
This value specifies the maximum number of billing events that the PTE records. As new billing events arrive the older events are deleted.
13. Select **Start** to start the Diameter service.
These next steps are necessary to specify the listening ports for your Diameter simulator in Services Gatekeeper.
14. Select the **Multi-home** sub-tab.
15. Copy the top port number or IP address to your system clipboard by clicking the Copy to Clipboard icon.
16. Select the **Server** tool in the upper right of the PTE GUI.
17. Select the **MBean** sub-tab.
18. Select **Connect**.
The MBean tree appears.
19. Select **wlng_nt_payment_px30#5.1 > Diameter_plugin_name > PaymentMBean**.
20. Paste the IP address from your clipboard into the **DestinationAddresses** attribute field.
21. Add a comma immediately after the IP address.
22. Return to the **Diameter Multi-home** sub-tab and copy the lower IP address to your clipboard.
23. Return to the **MBeans** sub-tab and paste the second IP address in the DestinationAddresses field after the comma without any blank spaces.
24. Select the green down arrow to make the change take effect.
You can add additional addresses up to a 255 character limit.
25. Select **Disconnect** in the MBean **Browser** tab, then select **Connect** again.
26. Select **wlng_nt_payment_px30#5.1 > Diameter_plugin_name > PaymentMBean > Connected**. The *Diameter_plugin_name* is the name given to the Diameter plug-in you created for testing.
27. Confirm that the **Connected** checkbox is checked, indicating that the PTE is now connected to the destination addresses.

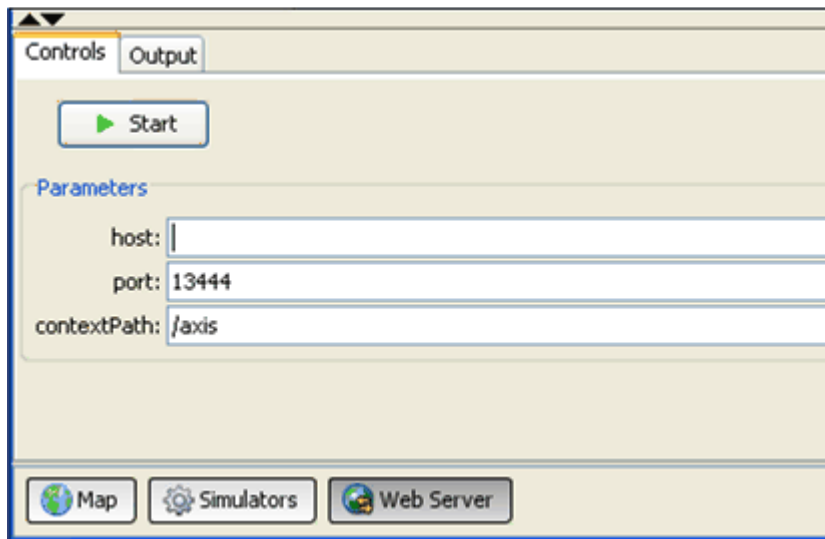
Configuring a Web Server for HTTP Testing

The **Web Server** button offers configuration and output for a simulated HTTP server. This simulator is required to run traffic over HTTP based protocols like MM7.

Set up and start a web server simulator:

1. Connect to a Services Gatekeeper implementation.
2. Click the **Web Server** button at the bottom of the Network Simulator panel. The **Web Server Controls** tab appears. [Figure 2-12](#) shows the Web Server Controls tab.

Figure 2–12 The Web Server Controls Tab



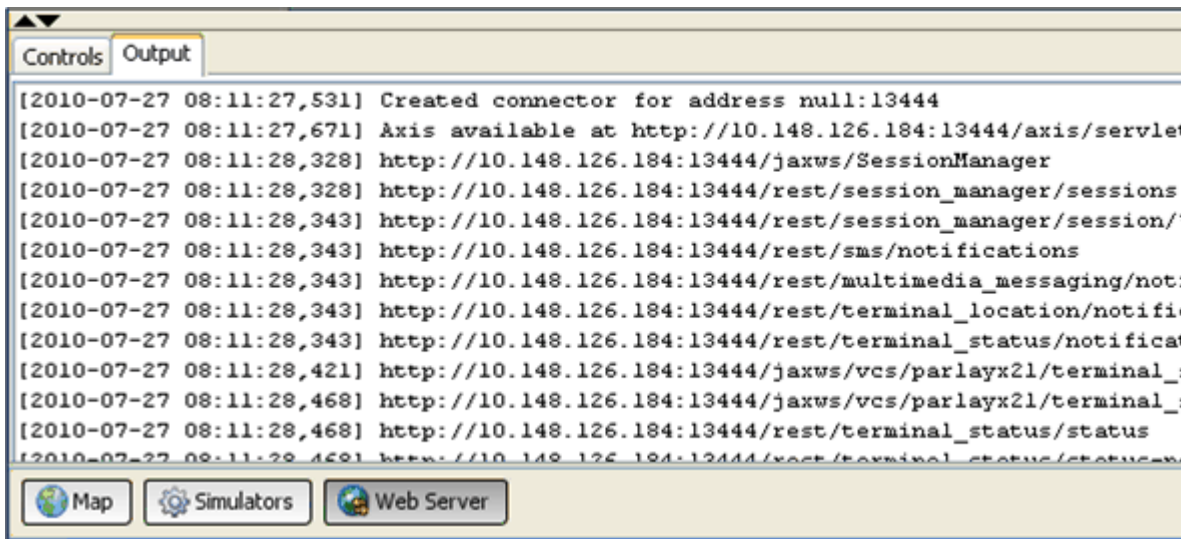
3. Enter the **host** of the web server to use.
4. Enter a **port** for the web server to use.
5. Enter **contextPath** (/axis by default) for the web server to use.
6. Click **Start** to start your web server.

The web server starts.

7. Click the **Output** tab and check for web server messages.

Figure 2–13 shows an example **Controls** tab populated with web server messages.

Figure 2–13 The Web Server Output tab



Configuring and Testing Service-Level Agreements

SLAs are XML files that you use to control your partners' access to the services that Services Gatekeeper manages. For information on SLAs, see *Services Gatekeeper Accounts and SLAs Guide*.

To help you manage SLAs, the PTE includes an SLA browser, which displays SLAs, and an SLA editor, which you use to change tags and validation. In a test environment, changing details multiple times for various iterations of testing is common, and these tools free you to focus on setting values.

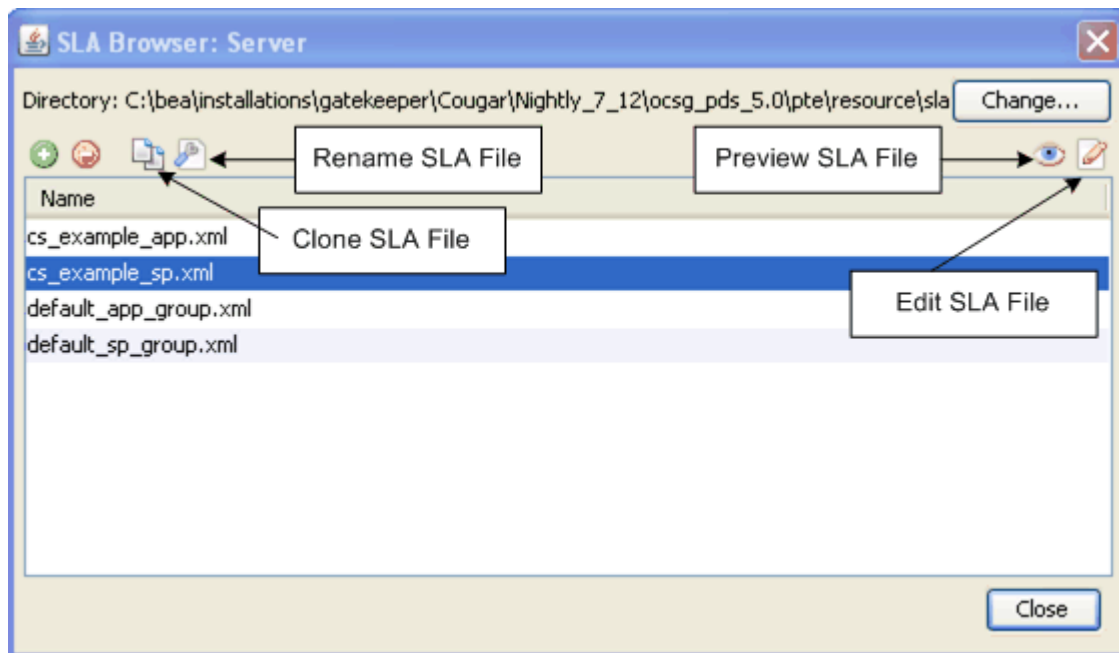
To access the SLA Manager:

1. Connect to a Services Gatekeeper implementation.
2. In the tools selection panel, click **Server**
3. Click the **MBean** sub-tab.
4. Click **Connect**.
5. Pull down the **Tools** menu from the top of the PTE, and select **SLA Manager**.

The SLA Manager appears displaying the example SLAs available in the default SLA directory: *Middleware_home/ocsg_pte/resource/sla* where *Middleware_home* is the middleware home directory where you installed Services Gatekeeper.

Figure 2-14 shows an example **SLA Browser** with SLAs available and the various **SLA Browser** tools highlighted.

Figure 2-14 The SLA Browser

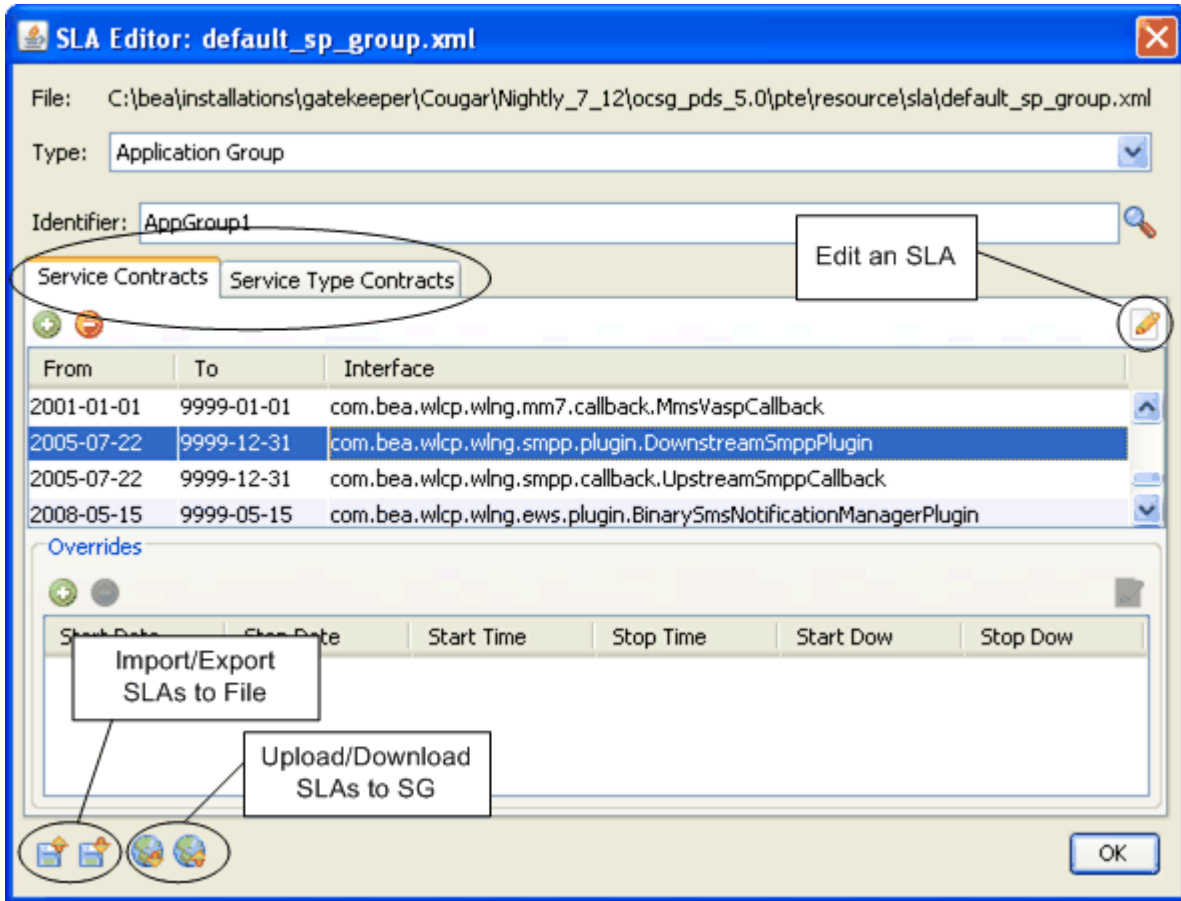


The example SLAs model the various options for SLA construction. To display a different SLA file, click **Change** and select it using the file browser that appears. To create a new SLA, click the plus button in the upper left corner of the **SLA Browser** window. You can also delete or rename SLAs from the same area.

To view a SLA XML file, select it, then click the eye icon in the upper right corner of the pane. To edit an SLA, select it and click the pencil icon in the upper right corner.

The **SLA Editor** window appear. [Figure 2–15](#) shows the **SLA Editor** window with its controls highlighted.

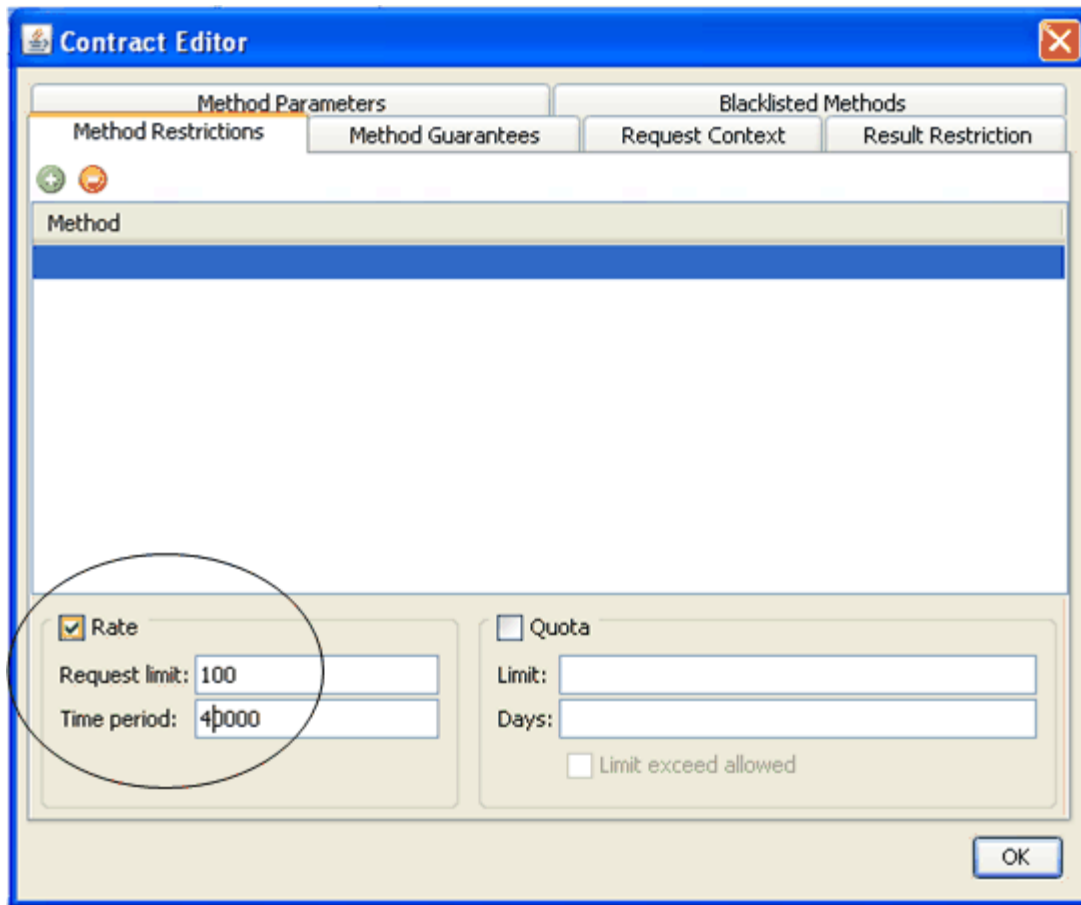
Figure 2–15 The SLA Editor



Select the **SLA Type** from the dropdown menu and specify the group identifier. Use the import and export icons on the bottom left to import an SLA from the file system or to save changes. Use the upload and download icons on the bottom left to upload the SLAs to, or download them from Services Gatekeeper.

You edit **Service Contracts** and **Service Type Contracts** using the tabs in the upper window, and edit any **Overrides** in the lower. To edit a service contract, click it and the **Contract Editor** window appears. [Figure 2–16](#) shows the **Contract Editor** window and the elements available to add or change.

Figure 2–16 The SLA Contract Editor



The editable tags appear as tabs at the top of the window. Make any changes and click **OK** to save them and close the window. For more information on these tags, see “Customizing SLA Behavior for a Service Provider or Application” in *Services Gatekeeper Portal Developer’s Guide*.

Figure 2–16 shows a new rate method restriction being added to `SLA_test1.xml`. Click **OK** to save your changes.

Click **OK** once again (on Figure 2–15) to finish.

To preview your edits in XML format, click the Eye icon at the top left of the SLA Browser (Figure 2–14). The **Preview SLA** window opens. See Figure 2–17.

Figure 2–17 The Preview SLA window



The Method Restriction rate limit that was added in Figure 2–17 is shown outlined in red.

When you have completed your edits, click the **Close** button on the **SLA Browser** window.

Configuring and Testing Composed Service-Level Agreements

This section describes how to use the SLA Manager in the PTE to edit a composed service contract in Oracle Communications Services Gatekeeper.

The PTE provides a graphical interface to the SLA editor for creating and managing composed SLAs. When a user modifies a composed SLA using this editor the corresponding XML file is updated immediately.

The editor provides browsers that provide access to information needed to create and manage SLAs, such as the available identifiers, communication service names, and methods.

For information about composed service contracts, see “Structure of a Composed Service Contract” in *Services Gatekeeper Accounts and SLAs Guide*.

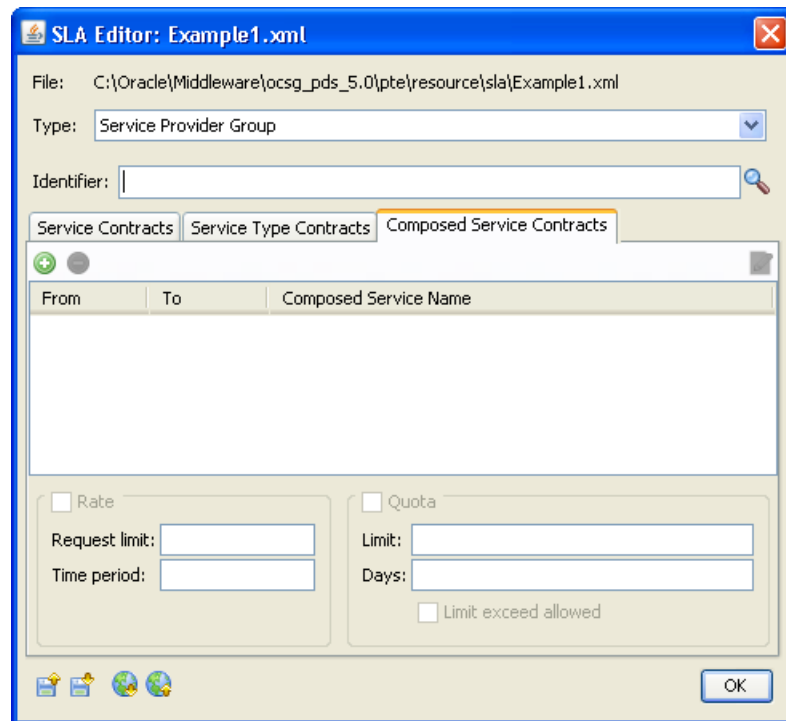
Starting the Composed SLA Editor

To start the Composed SLA Editor:

1. In the PTE, connect to Services Gatekeeper and start the SLA Manager.
2. In the SLA Manager, select the **Composed Service Contracts** tab.

The Composed SLA Editor appears.

Figure 2–18 Composed Service Editor



Downloading an Existing Composed SLA

If you have an existing Composed SLA that you want to edit, you can download it from the Services Gatekeeper server.

To download an existing SLA:

1. With the **Composed Service Contract** tab selected, select the type of SLA from the **Type** menu.
2. Enter the identifier for the SLA in the **Identifier** field. You can click the search icon to browse for the identifier:



3. Select the SLA to download.
4. Click download icon at the bottom of the editor to download the SLA:



The **Selector** box appears.

5. Select the contract that you want to download from the **Selector** box.
6. Click **OK**.

Creating a New Composed SLA

To create a new Composed SLA:

1. With the **Composed Service Contract** tab selected, select the type of SLA that you want to create from the **Type** menu.
2. Assign a unique identifier for the composed SLA in the **Identifier** field.
3. Click the add icon to create a new SLA:



The new SLA appears in the editor.

4. Add the start and end dates and the composed service name.

See "[Composed Service Name and Start/End Dates](#)" for instructions on how to assign a name and dates to the composed service contract.

5. Click **OK**.

Editing an SLA

You can edit the following values in a new or previously existing contract:

- [Composed Service Name and Start/End Dates](#)
- [Rates and Quotas](#)
- [Communication Services and Methods](#)

For information about what these values represent, see "Defining Service Provider Group and Application Group SLAs" in *Services Gatekeeper Portal Developer's Guide*. The following sections describe only the mechanics of modifying these values in the editor.

Composed Service Name and Start/End Dates To change the **Composed Service Name**, start date (**From** field), or end date (**To** field):

1. In the composed service editor with the **Composed Service Contract** tab selected, click the field that you want to modify.
2. Over-write the existing value with the new value.

If you are editing the **Composed Service Name** field, make sure to enter a unique name for the composed service.

3. Click **OK**.

Rates and Quotas To edit the rate or quota at the bottom of the editor, with the SLA selected:

1. Check the check box for the rate or quota to be enforced by the SLA.
2. Enter the values for the rate or quota to be applied.
3. Click **OK**.

Communication Services and Methods This section describes how to add communication services to the composed service and how to specify the methods that the composed service contract applies.

The methods portion is optional. If no methods are specified, the enforcements apply to all of the communication services's methods. If at least one method is specified, only the specified method or methods participate in the composed service.

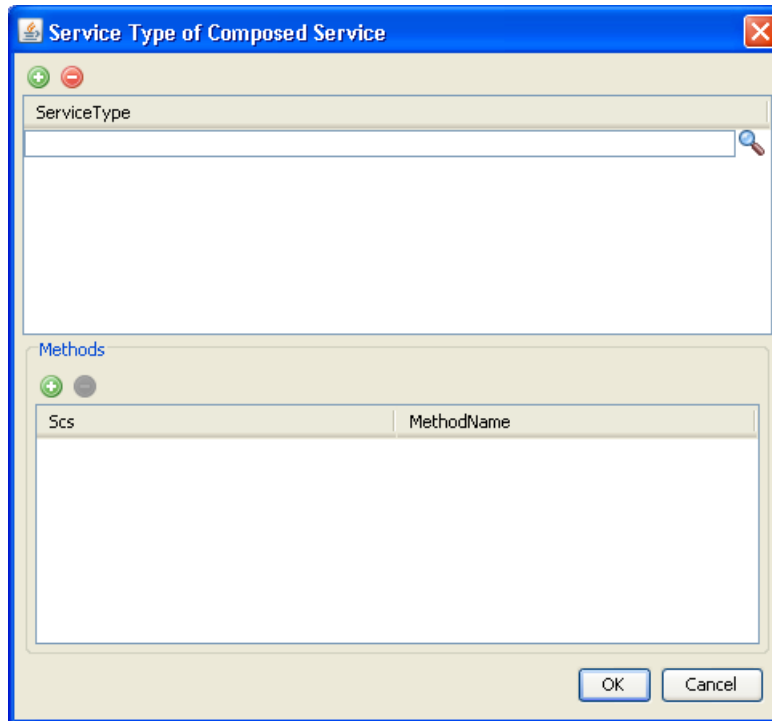
Adding Communication Services To add the communication services that define the composed service contract:

1. Select the Composed SLA in the editor.
2. Click the edit icon:



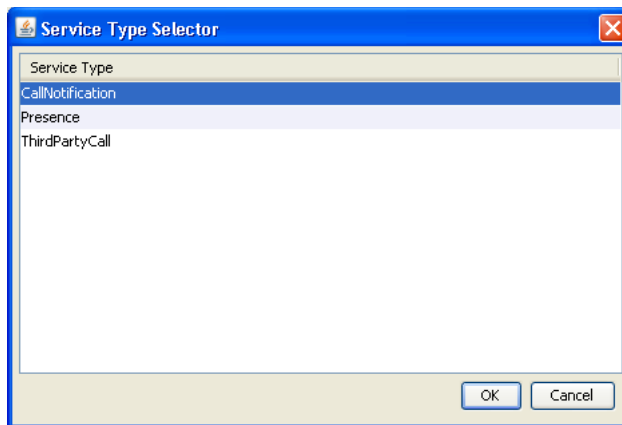
The **Service Type of Composed Service** box appears. Use the upper pane to add communication services.

Figure 2–19 Service Type of Composed Service Box



3. For every communication service that you want to add to the composed service, do the following:
 - a. In the **Service Type of Composed Service** pane, click the add button to add a service.
 - b. Click the search button to load the list of available communication services. The **Service Type Selector** list appears.

Figure 2–20 Service Type Selector



- c. In the **Service Type Selector** box, select a service type.
- d. Click **OK**.

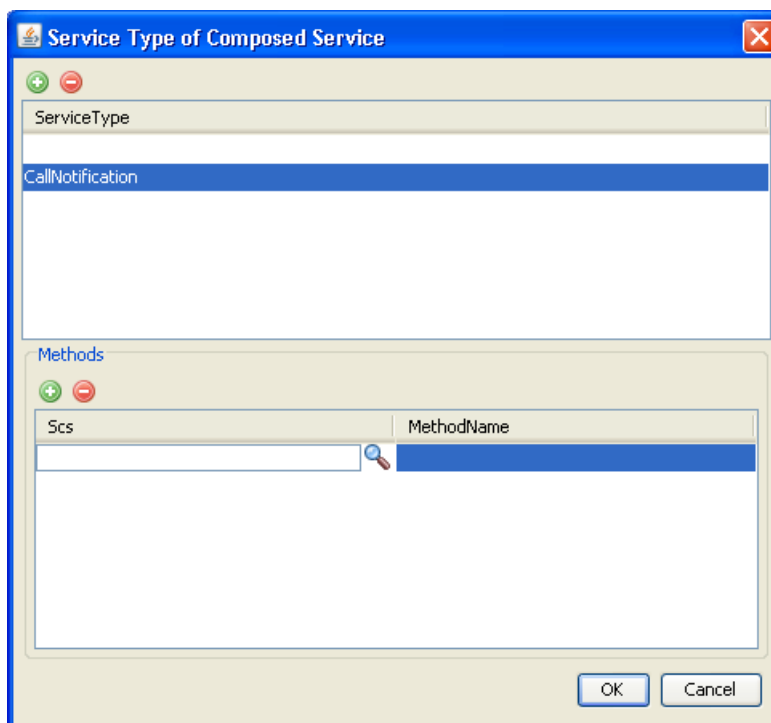
Adding Methods If you want to limit SLA enforcement to specific methods of a communication service, use the lower pane of the **Service Type of Composed Service** box.

To specify a method to which the composed service contract enforcements are applied:

1. With the communication service selected in the upper pane of the **Service Type of Composed Service** box, click the add button in the lower pane.

A row for the method appears in the lower **Scs** pane.

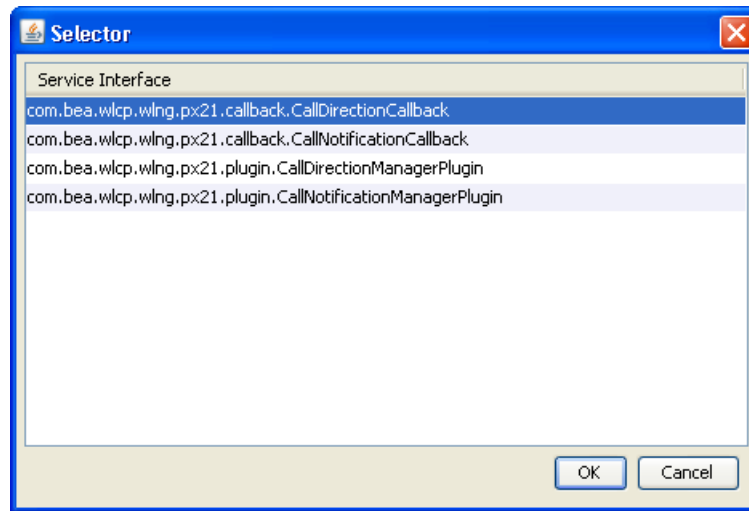
Figure 2–21 Service Type of Composed Service Box with SCS Search Button



2. Click the search button.

The Scs Selector box appears.

Figure 2–22 Scs Selector



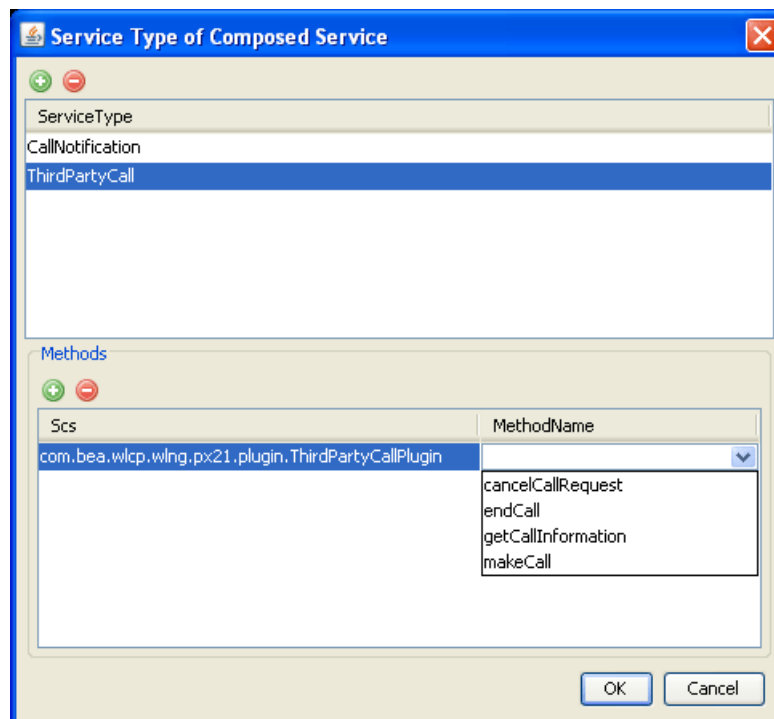
3. Select the service interface.
4. Click **OK**.

The service interface appears in the lower pane of the Service Type of Composed Service box.

5. Click the Method Name area next to the selected Scs to display the available methods for the Scs.

A menu of the available methods appears.

Figure 2–23 Service Type of Composed Service Box with Method Menu



6. Select a method name from the menu.
7. Click **OK**

Repeat this procedure for every method that you want to add to the SLA enforcement.

Upload an Edited Composed SLA

To upload the edited SLA:

1. Select the SLA in the editor and click the upload icon:



An alert asks whether the composed SLA is geographically redundant or local.

2. Respond to the alert by clicking the **Geo-redundant** or **Local** button.

A local SLA is enforced locally in the network tier cluster into which it is loaded. A geographically redundant SLA is loaded to and enforced across all network tier clusters in a geographically redundant configuration.

The composed SLA is uploaded.

Monitoring Budget Use

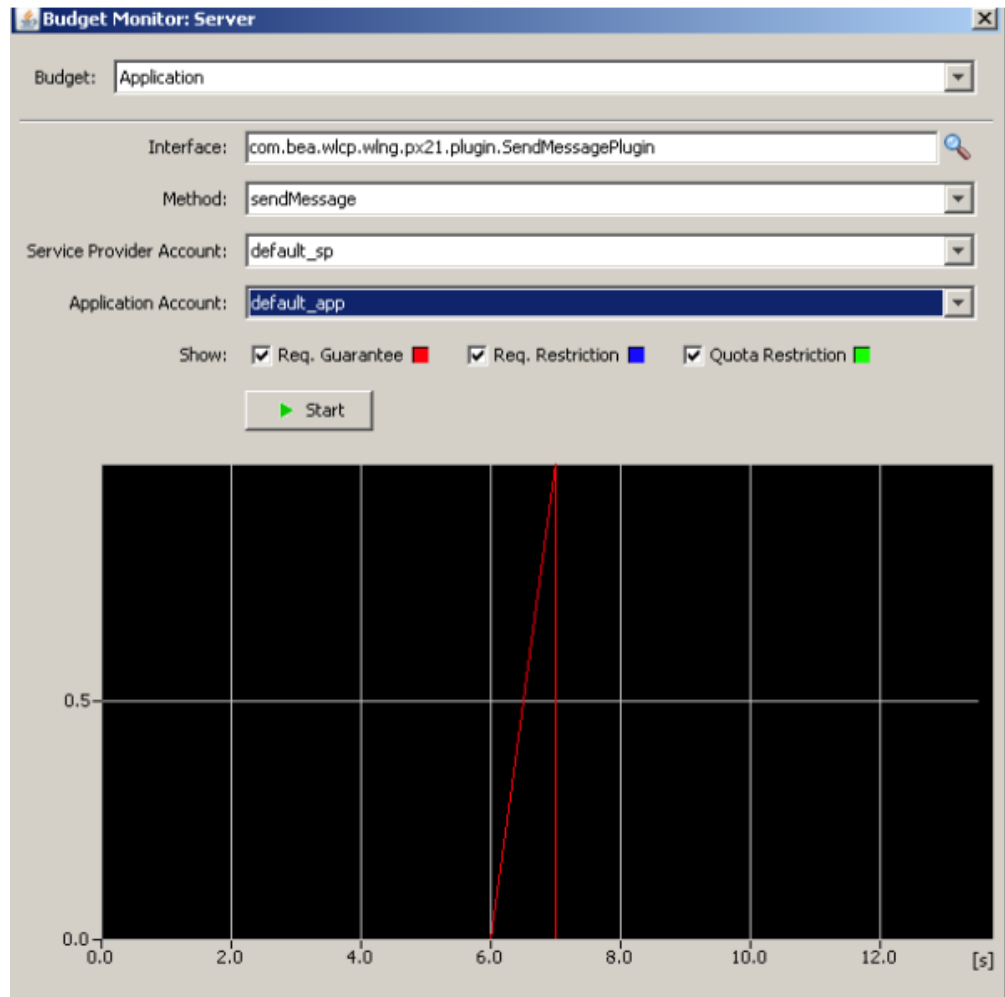
Budgets measure the level of access an application has to Services Gatekeeper over time. For information on budgets, see “Managing and Configuring Budgets” in *Services Gatekeeper System Administrator’s Guide*.

You must create instances of certain communication services to use them in Services Gatekeeper. For these communication services, only service types that have been instantiated show up in the **Selector** panel.

To monitor budget use:

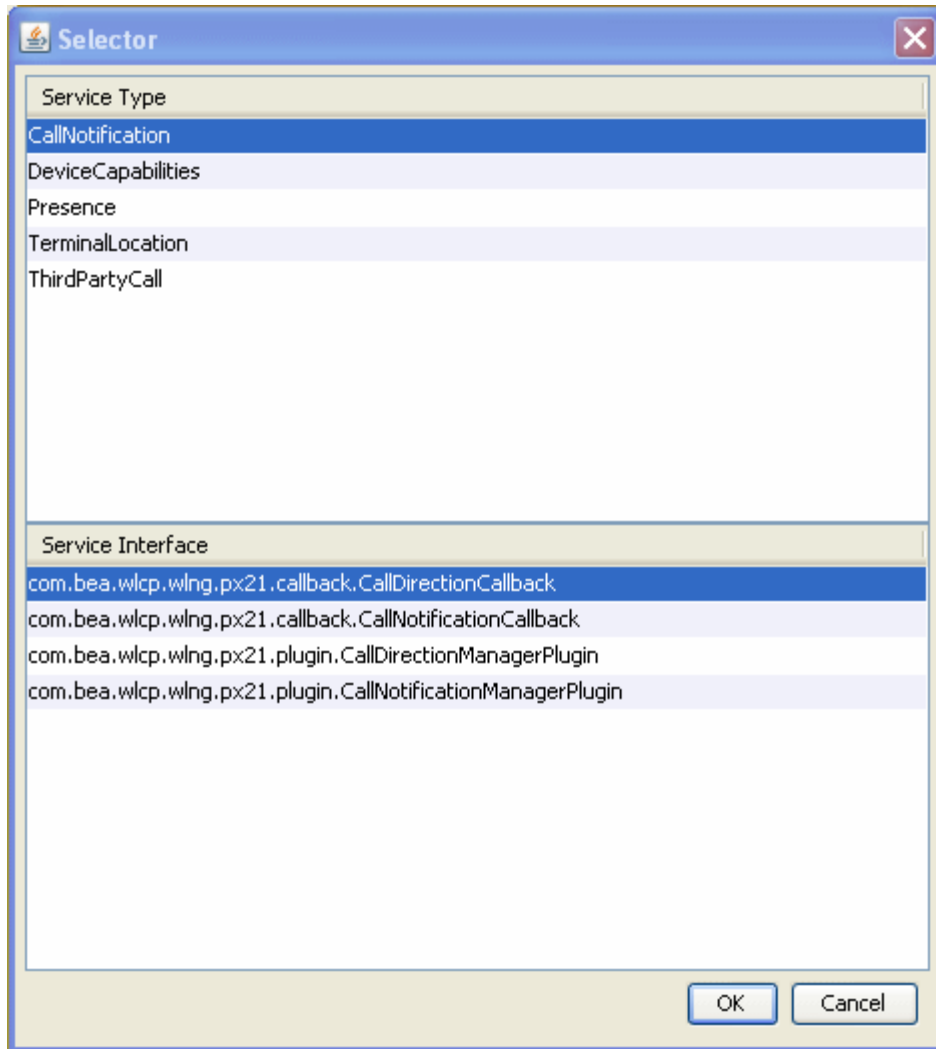
1. Connect to a Services Gatekeeper implementation.
2. In the Tools Selection panel, select the **Server**.
3. From menu bar at the top left of the PTE GUI, select the **Tools** menu, and then **Budget Monitor**. The Budget Monitor window appears. [Figure 2-24](#) shows the **Budget Monitor** window.

Figure 2–24 The Budget Monitor window



4. From the **Budget** drop-down list, select a type of budget to monitor.
5. If you selected an **Application** or **Service Provider**, do the following:
 - a. The **Selector** window appears. [Figure 2–25](#) shows the Selector window populated with service types and service interfaces.
 - b. Select the **Service Type** and a **Service Interface** to monitor.
 - c. Click **OK**. The **Selector** window closes and the **Budget Monitor** window reappears.

Figure 2–25 The Selector Window



6. Select a **Method** and its associated **Service Provider Account** from the lists.
7. Click **Start**.

The X-axis indicates time in seconds and the Y-axis the amount of budget that is available. The number of units in the Y-axis depends on values assigned in the SLA for the selected budget. Rates, quotas, and restrictions can all be mapped.

Adding and Testing Custom Client Modules

This chapter explains how to extend the Oracle Communications Services Gatekeeper Platform Test Environment (PTE) by adding and testing custom client modules. Client modules serve as the application-facing interfaces for communication services. Client modules are displayed in the PTE GUI when you select the **Clients** item in the tools selection panel. An example unit test is included with this release that you can use as a model for your own customizations.

Adding Custom PTE Client Modules to the PTE

For instructions on how to create and deploy new client modules (application-facing), network protocol simulators (network-facing), and communication services, see "Extending the ATE and the PTE" in *Services Gatekeeper Extension Developer's Guide*.

Testing New or Changed Client Modules

This section explains how to verify that your new or modified client modules are functioning correctly.

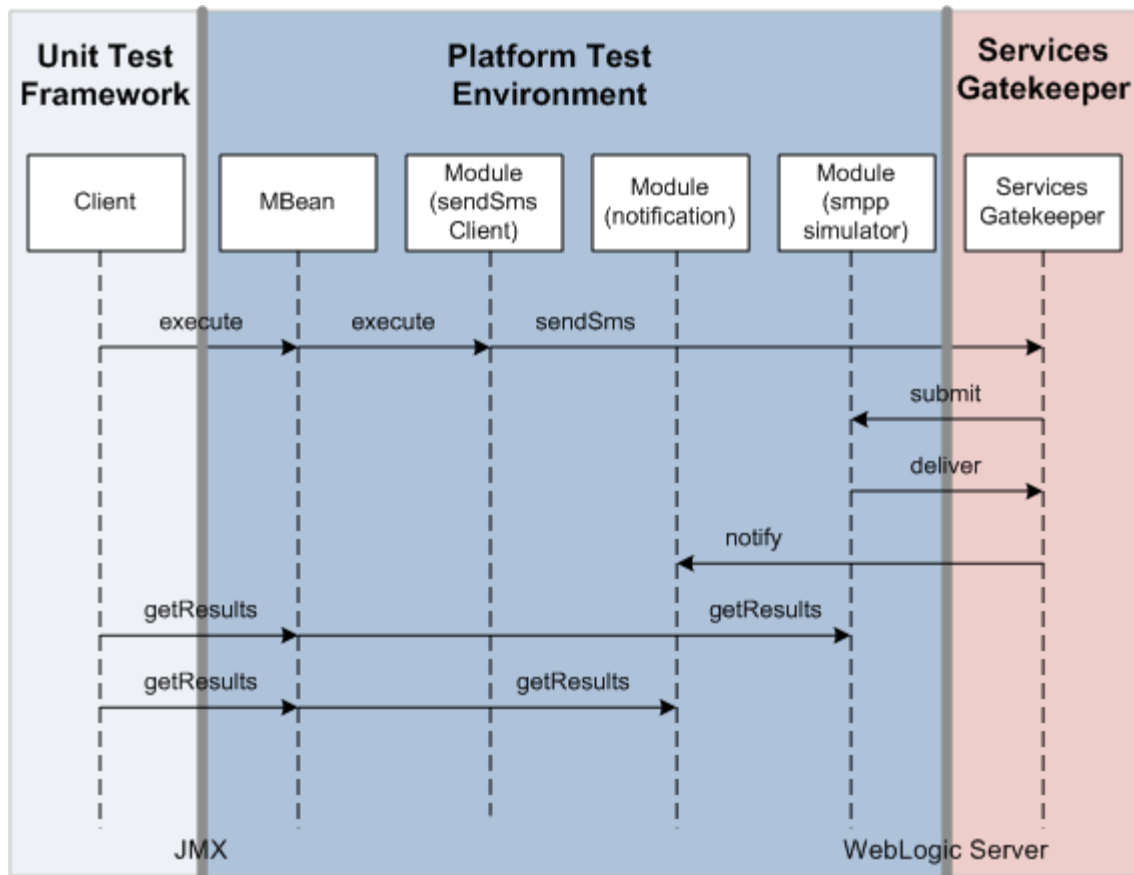
Understanding Communication Service Unit Tests

Unit tests are an essential part of creating or changing a Services Gatekeeper communication service. You add known data to the system, run tests on it, and confirm that the test results are what you expect, all programmatically.

You use the Unit Test Framework to create PTE unit tests. Its main tools include:

- The `OCSGBaseTestCase` abstract class (located in `Middleware_home/ocsg_pds/lib/wlng/pte_api.jar` where `Middleware_home` is the middleware home where Services Gatekeeper is installed) manages the mechanics of using JMX and JMS to connect to the PTE.
- The `test.properties` file (located in `Middleware_home/ocsg_pds/lib/wlng/pte_api.jar` where `Middleware_home` is the middleware home where Services Gatekeeper is installed) defines the commonly changed properties of the test.

Figure 3-1 An Example SMS Unit Test Workflow



The workflow of the SMS unit test is as follows:

1. The test client calls `execute` on the PTE Module Management MBean. The mechanics of the JMX call are taken care of by the base class.
2. The MBean calls `execute` on the specified module; in this case, `sendSMS`. This request includes a request for delivery receipts.
3. The `sendSMS` module sends the request to Services Gatekeeper.
4. Services Gatekeeper processes the request and submits it to the network simulator module; in this case, the SMPP module.
5. The simulator module returns a delivery receipt to Services Gatekeeper.
6. Services Gatekeeper sends the receipt on to the Notification module (which represents the client web Service implementation).
7. The test client retrieves the result of Services Gatekeeper's `submit` from the SMPP simulator.
8. The test client retrieves the delivery receipt from the Notification module.

Understanding the Example Unit Test

To help you understand how unit tests work, PTE includes an example unit test (`TestSendData.java`), which tests an example communication service using the example clients and simulator. The default location for this file is `Middleware_home/ocsg_pds/example/unit_test/src/oracle/ocsg/et/example/TestSendData.java`

where *Middleware_home* is the middleware home where Services Gatekeeper is installed. [Example 3-1](#) shows the `TestSendData.java` file example communication service test.

Example 3-1 A Unit Test for the Example Communication Service

```
package oracle.ocsg.et.example;
import oracle.ocsg.ptc.api.OCSGBaseTestCase;
import java.util.List;
import java.util.Map;
/**
 * This class illustrates how to use the Unit Test Framework to
 * test the Communication Service Example. A few things are assumed before
 * running this class:
 * - the OCSG should be running and configured properly
 * - the CS example should be deployed and ready
 * Note: this example uses also the wlngJmx to be able to access the OCSG
 * MBeans to ask the CS example plugin to connect to the Netex simulator.
 */
public class TestSendData extends OCSGBaseTestCase {
    private static final String SEND_DATA_MBEAN =
"com.bea.wlcp.wlng.ptc:group=traffic,name=example_application_initiated";
    private static final String NETWORK_TRIGGERED_MBEAN =
"com.bea.wlcp.wlng.ptc:group=traffic,name=example_network_triggered";
    private static final String NOTIF_MANAGER_MBEAN =
"com.bea.wlcp.wlng.ptc:group=client,name=example_notif_manager";
    private static final String NOTIF_MBEAN =
"com.bea.wlcp.wlng.ptc:group=client,name=example_notif";
    private static final String NETEX_SIMULATOR_MBEAN =
"com.bea.wlcp.wlng.ptc:group=netex,name=example_simulator";

    // Make sure the InstanceName correspond to the instance name provided
    // in the PluginManager Mbean when creating the plugin instance.
    private static final String EXAMPLE_PLUGIN_MBEAN =
        "com.bea.wlcp.wlng:AppName=example_enabler#4.0," +
        "InstanceName=example_1," +
"Type=com.acompany.plugin.example.netex.management.ExampleMBean";
    public TestSendData() throws Exception {
    }

    @Override

    protected void setUp() throws Exception {
        super.setUp();
        ocsgJmx.open("localhost", 8001, "weblogic", "weblogic");
        start(NETEX_SIMULATOR_MBEAN);
    }

    @Override
    protected void tearDown() throws Exception {
        ocsgJmx.close();
        stop(NETEX_SIMULATOR_MBEAN);
        super.tearDown();
    }

    public void testSendData() throws Exception {
        assertTrue(isRunning(NETEX_SIMULATOR_MBEAN));

        resetStatistics(NETEX_SIMULATOR_MBEAN);
    }
}
```

```

        ocsGJmx.invokeOperation(EXAMPLE_PLUGIN_MBEAN, "connect");
        String data = "Hello at " + System.currentTimeMillis();
        String to = "tel:1234";
        putParameter(SEND_DATA_MBEAN, "url",
"http://localhost:8001/example/SendData");
        putParameter(SEND_DATA_MBEAN, "data.data", data);
        putParameter(SEND_DATA_MBEAN, "data.address", to);

        start(SESSION_MBEAN);
        assertTrue(isRunning(SESSION_MBEAN));
        execute(SEND_DATA_MBEAN);
        Thread.sleep(2000);

        stop(SESSION_MBEAN);

        Map<String,String> stats = listAllStatistics(NETEX_SIMULATOR_MBEAN);
        System.out.println("Simulator statistics: "+stats);
        assertEquals("MessageReceived", "1", stats.get("MessageReceived"));
        assertEquals("MessageSent", "0", stats.get("MessageSent"));

public void testSendNetworkTriggeredData() throws Exception {
        String data = "Hello at " + System.currentTimeMillis();
        String from = "tel:1234";
        String to = "tel:7878";
        String correlator = "1234567890";
        assertTrue(isRunning(NETEX_SIMULATOR_MBEAN));
        resetStatistics(NETEX_SIMULATOR_MBEAN);
        ocsGJmx.invokeOperation(EXAMPLE_PLUGIN_MBEAN, "connect");

        start(SESSION_MBEAN);
        assertTrue(isRunning(SESSION_MBEAN));
        putParameter(NOTIF_MANAGER_MBEAN, "url",
"http://localhost:8001/example/NotificationManager");
        putParameter(NOTIF_MANAGER_MBEAN, "start.address", "tel:7878");
        putParameter(NOTIF_MANAGER_MBEAN, "start.correlator", correlator);
        putParameter(NOTIF_MANAGER_MBEAN, "start.endpoint",
"http://localhost:13444/jaxws/NetexNotification");
        putParameter(NOTIF_MANAGER_MBEAN, "stop.correlator", correlator);
        start(NOTIF_MANAGER_MBEAN);

        start(NOTIF_MBEAN);
        putParameter(NETWORK_TRIGGERED_MBEAN, "data", data);
        putParameter(NETWORK_TRIGGERED_MBEAN, "fromAddress", from);
        putParameter(NETWORK_TRIGGERED_MBEAN, "toAddress", to);

        clearResults(NOTIF_MBEAN);

        execute(NETWORK_TRIGGERED_MBEAN);

        Thread.sleep(2000);

        stop(NOTIF_MBEAN);
        stop(NOTIF_MANAGER_MBEAN);
        stop(SESSION_MBEAN);
}

        Map<String,String> stats = listAllStatistics(NETEX_SIMULATOR_MBEAN);
        System.out.println("Simulator statistics: "+stats);
        assertEquals("MessageReceived", "0", stats.get("MessageReceived"));
        assertEquals("MessageSent", "1", stats.get("MessageSent"));

```

```
List<Map<String,String>> results = listAllResults(NOTIF_MBEAN);
System.out.println("Notification results: "+results);
assertEquals("Correlator", correlator, results.get(0).get("Correlator"));
assertEquals("From Address", from, results.get(0).get("From Address"));
assertEquals("Data", data, results.get(0).get("Data"));

}
```

Creating and Running a New Unit Test

To create and run a PTE unit test:

Note: The PTE should be running in console (non-GUI) mode when you run your test. See ["Starting the PTE"](#) for instructions on starting the PTE in console mode.

1. Create any necessary client or simulator modules for the PTE using the required SPI and XML configuration files. See ["Adding Custom PTE Client Modules to the PTE"](#) for information.
2. Implement a test class based on the `OCSGBaseTestCase` abstract class.
3. Add the test class to the PTE. For information, see ["Extending the ATE and the PTE"](#) in *Services Gatekeeper Extension Developer's Guide*.
4. Start the PTE and make sure the modules are correctly loaded. For information on starting PTE, see ["Starting the PTE"](#). If your new module is correctly loaded, it appears in the PTE GUI.
5. Run the test class. See ["Using the PTE to Test Services Gatekeeper"](#) for information.

[Figure 3–1](#) shows the workflow of a typical unit test; in this case an SMS communication service unit test.

