Oracle® Communications Network Charging and Control

Open Services Development User's and Technical Guide

Release 15.2

January 2026

Copyright

Contents

About This Document

Scope

The scope of this document includes all functionality a user must know in order to effectively operate the Open Services Development application. It does not include detailed design of the service.

Audience

This guide is written primarily for Open Services Development (OSD) administrators. However, the overview sections of the document are useful to anyone requiring an introduction.

Prerequisites

A solid understanding of UNIX and a familiarity with IN concepts are an essential prerequisite for safely using the information contained in this technical guide. Attempting to install, remove, configure or otherwise alter the described system without the appropriate background skills, could cause damage to the system; including temporary or permanent incorrect operation, loss of service, and may render your system beyond recovery.

Although it is not a prerequisite to using this guide, familiarity with the target platform would be an advantage.

This manual describes system tasks that should only be carried out by suitably trained operators.

Related Documents

The following documents are related to this document:

- *Advanced Control Services Technical Guide*
- *Control Plan Editor User's Guide*
- *Notification Gateway Technical Guide*

Document Conventions

Typographical Conventions

The following terms and typographical conventions are used in the Oracle Communications Network Charging and Control (NCC) documentation.

Formatting Convention	Type of Information
Special Bold	Items you must select, such as names of tabs. Names of database tables and fields.
Italics	Name of a document, chapter, topic or other publication. Emphasis within text.
Button	The name of a button to click or a key to press. **Example:** To close the window, either click **Close**, or press **Esc**.
Key+Key	Key combinations for which the user must press and hold down one key and then press another. Example: **Ctrl+P** or **Alt+F4**.
`Monospace`	Examples of code or standard output.
`Monospace Bold`	Text that you must enter.
variable	Used to indicate variables or text that should be replaced with an actual value.
menu option > menu option >	Used to indicate the cascading menu option to be selected. Example: **Operator Functions > Report Functions**
hypertext link	Used to indicate a hypertext link.

Specialized terms and acronyms are defined in the glossary at the end of this guide.

Chapter 1

System Overview

Overview

Introduction

This chapter provides a high-level overview of the application. It explains the basic functionality of the system and lists the main components.

It is not intended to advise on any specific Oracle Communications Network Charging and Control (NCC) network or service implications of the product.

In this Chapter

This chapter contains the following topics.

What is Open Services Development?

Introduction

Open Services Development (OSD) enables third parties to submit Web Services Description Language (WSDL) files that invoke control plans.

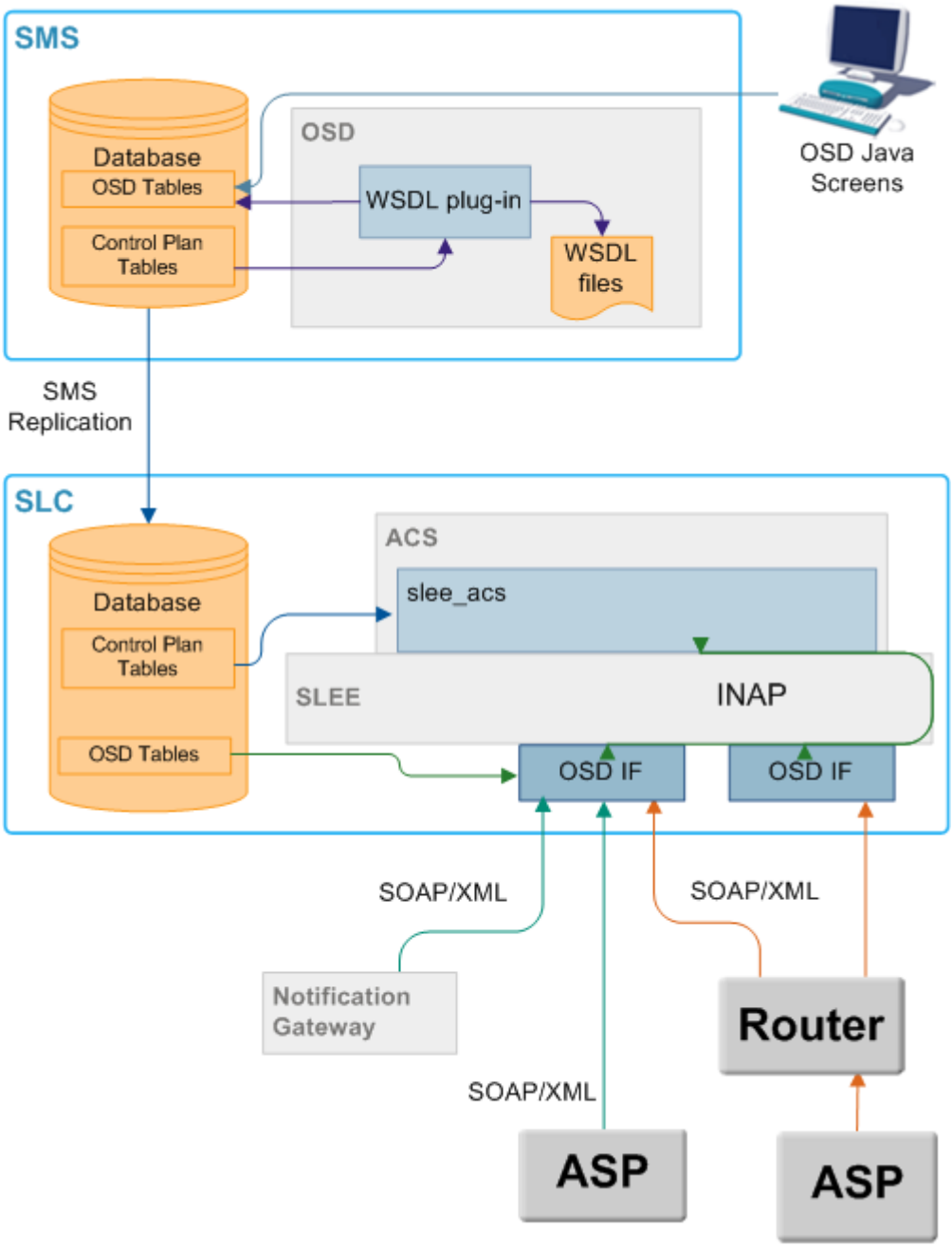
About the Notification Gateway

The NCC Notification Gateway (NGW) receives notifications from Oracle Communications Billing and Revenue Management (BRM Elastic Charging Engine (ECE) and transforms those notifications, via the OSD osdInterface, into messages that NCC can pass to subscribers.

See *Notification Gateway Technical Guide* for more information about NGW.

Architecture

This diagram shows the OSD architecture within a Oracle Communications Network Charging and Control environment.



Typical OSD Configuration Scenario

Introduction

This example is intended to show the nature of completing an OSD configuration, rather than an exact configuration.

OSD configuration process

Configuration for OSD is an iterative process of:

1. Complete **Service Provider** tab.
2. Complete **Operation Sets** tab.
3. Complete **Operation** tab.
4. Complete **Client ASP** tab.
5. If you will be using Notification Gateway to send notifications to subscribers from a third party, set the Notification Gateway username and password.
6. Create control plan for each operation.
7. Compile control plans.
8. Review WSDL through Operation screen (for each control plan).
9. Review WSDL through Operation Sets screen for the service supplier.

Example scenario

This scenario uses a control plan that copies data from the incoming Simple Object Access Protocol (SOAP) request to the outgoing SOAP result (this is not very useful but is a good illustration of how the software works).

- This copies from a short integer tag called ShortLO profile field in the incoming session data, which is mapped from the SOAP request, to a tag called ShortOPLO in the outgoing session data, which is mapped to the SOAP result.
- The incoming data is the number 12 in this scenario, see *Incoming SOAP message* (on page 11).
- The data is copied to ShortOPLO and the response returned through an out-going SOAP message.

To achieve this:

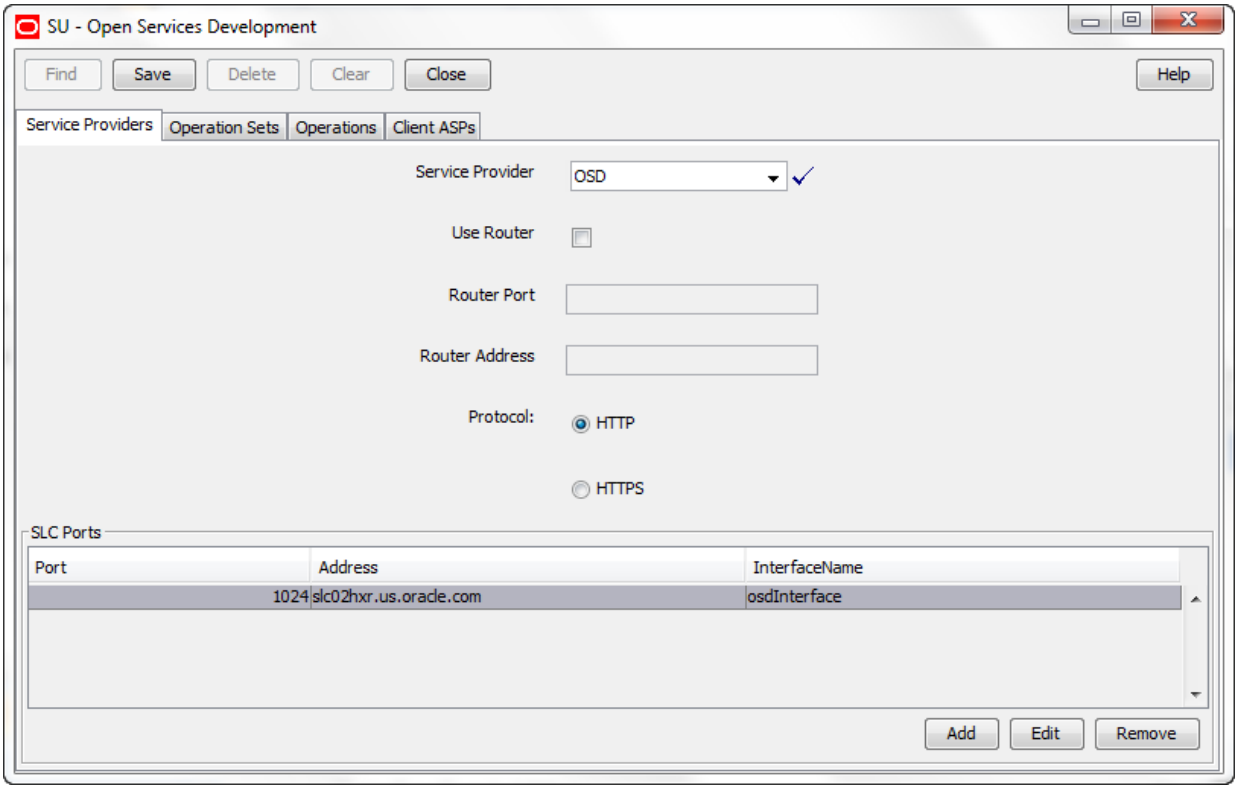
1. The OSD screens are configured as shown.
2. The control plan is created and compiled.
3. The generated WSDL is given to the third party.
4. The third party uses the WSDL by adding the relevant data, and sending the SOAP message to the OSD interface.
5. The OSD interface recognizes what control plan to invoke, see *Incoming SOAP message* (on page 11)).
6. The control plan copies the number and responds to the OSD interface.
7. The OSD interface returns a SOAP message to the third party, see *Outgoing SOAP message* (on page 11)).

Service Providers tab

Here is the **Service Providers** tab, configured for the scenario.

Notes:

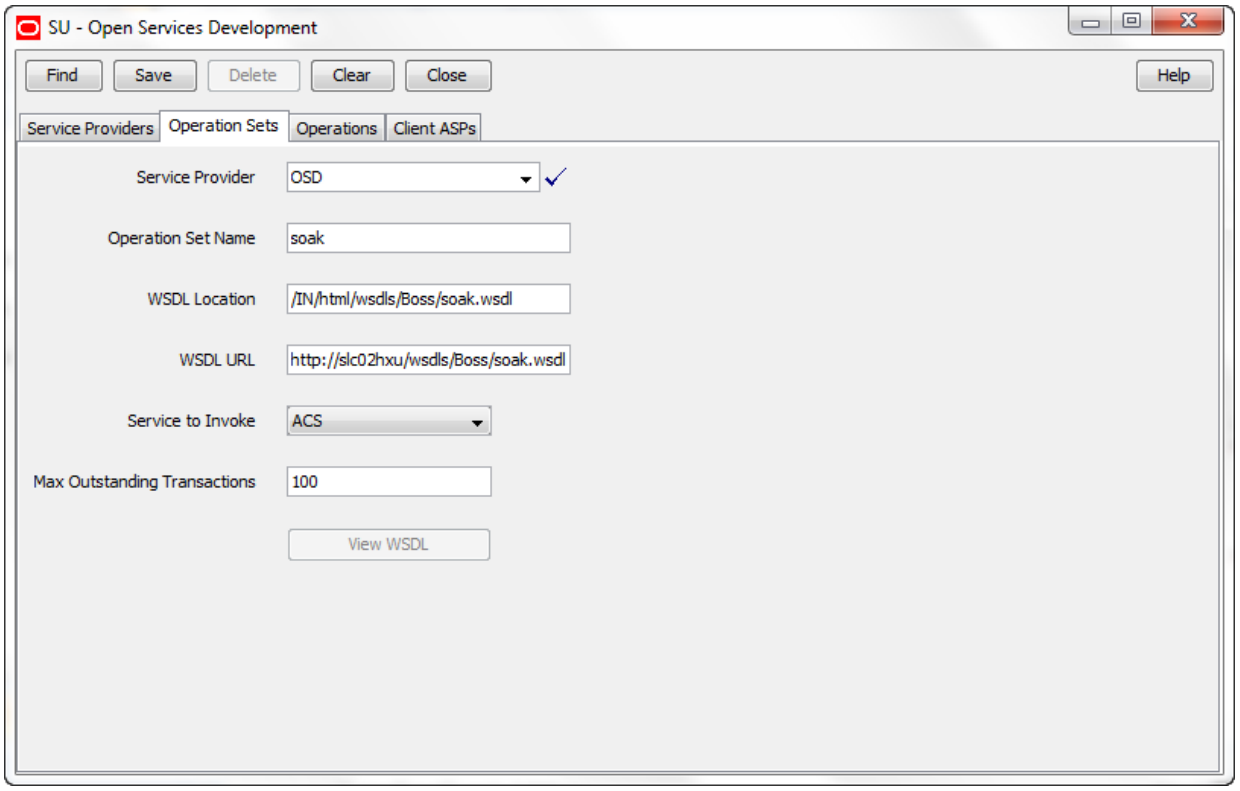
- The UAS port and address configured below must match the VWS **eserv.config** file triggering section address and port information.
- The InterfaceName must match the configured OSD interface running on the SLC SLEE.



Operation Sets tab

Here is the **Operation Sets** tab, configured for the scenario.

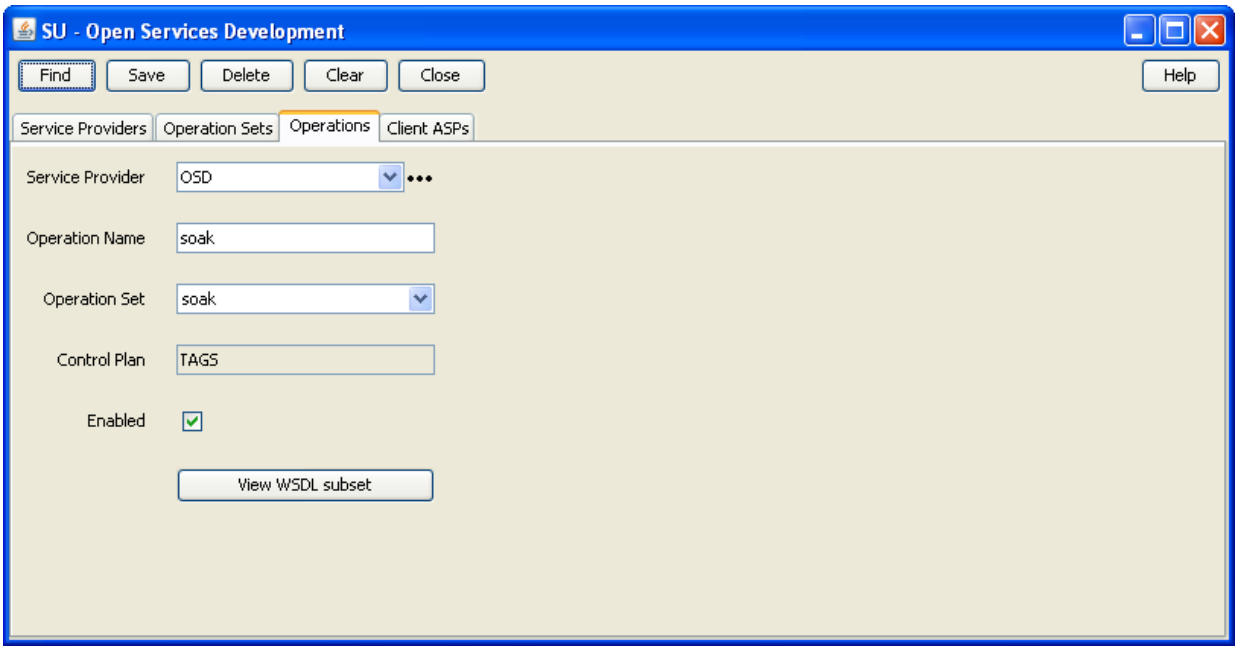
Note: The service to invoke is the capability that will be triggered on the SLC SLEE. For example, if the CCS service loader is used, this field will specify the CCS capability that will be triggered.



Operations tab

Here is the **Operations** tab, configured for the scenario.

Note: The control plan name is populated automatically when this operation is selected for WSDL generation during control plan compilation. The operation should only be enabled after a control plan WSDL has been generated.

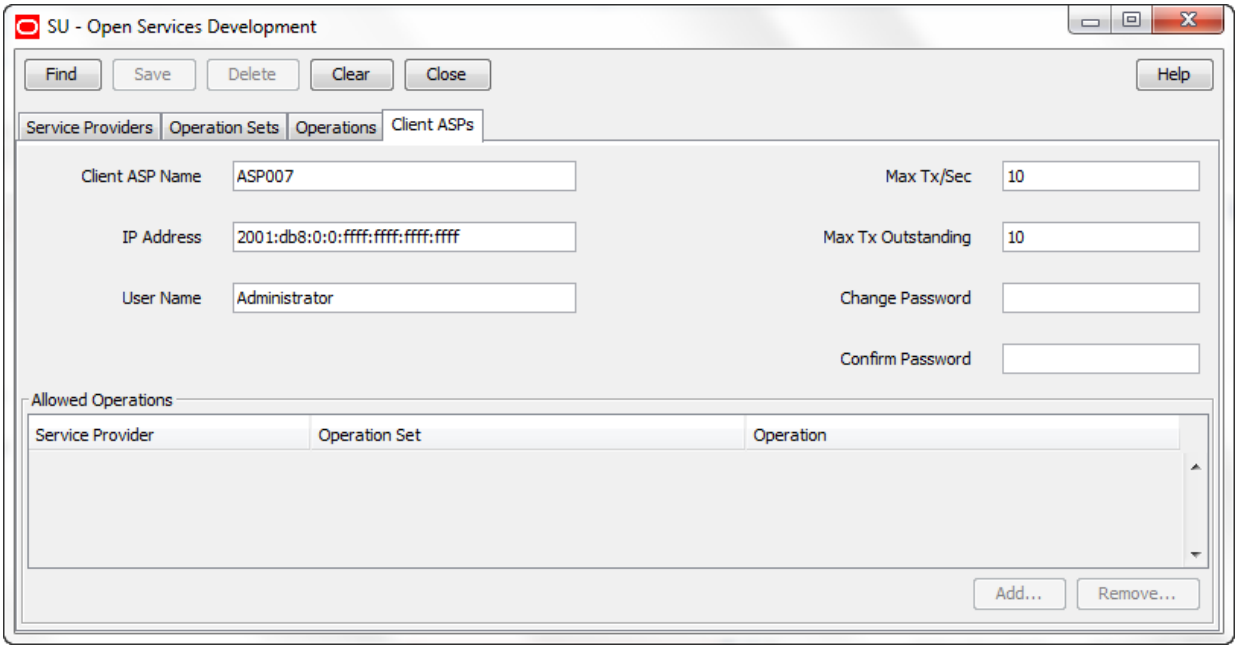


Client ASPs tab

Here is the **Client ASPs** tab, configured for the scenario.

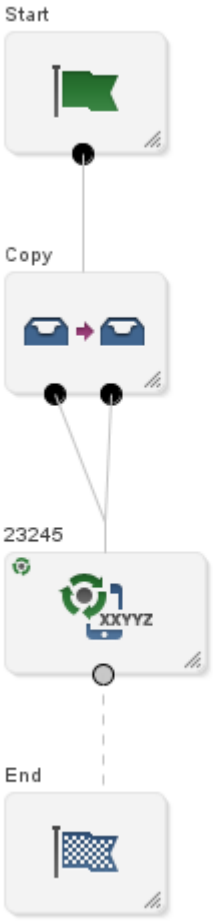
Notes:

- The client ASP name and IP address are the details of the VWS.
- The user name and password information configured must match the triggering operations overrides section of the **eserv.config** file on the VWS.



Control Plan

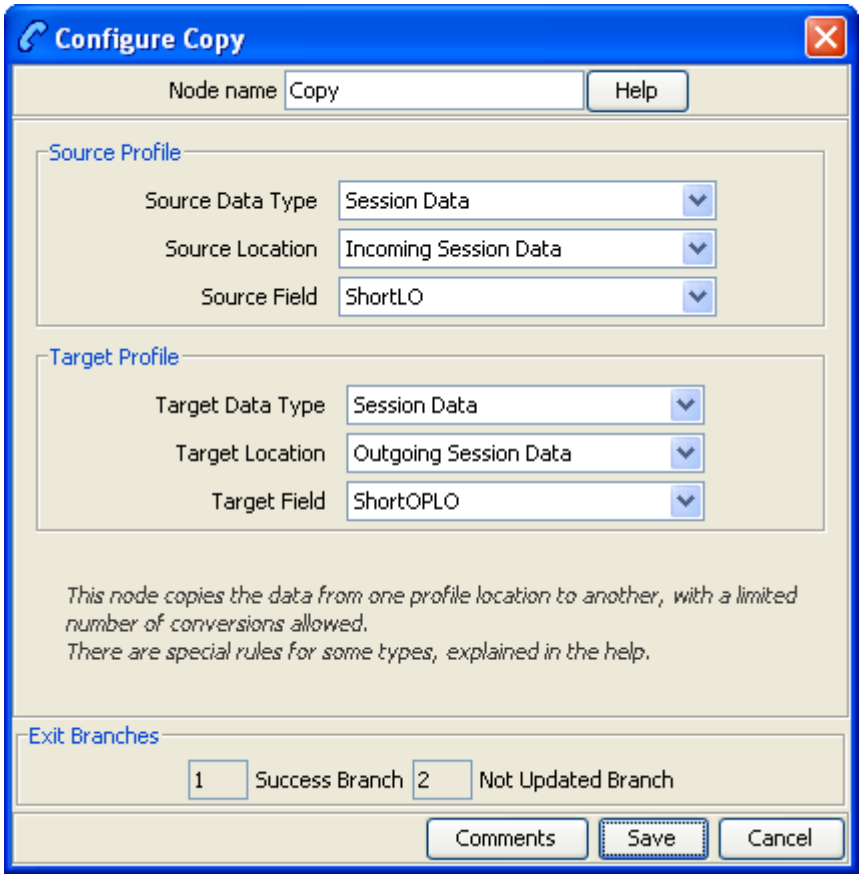
Here is the control plan for the scenario.



Important: An OSD control plan must contain an Unconditional Terminate or Unconditional Terminate to Pending feature node.

Copy Node Configuration

Here is the Copy node configuration detail for this scenario.



Control plan compilation

When the control plan is compiled, parameters are inserted into the WSDL using the OSD and Copy feature node configuration screens to provide the parameter names:

- "CC_Service_Number" – is the number to trigger the control plan which we configured in the **ACS Numbers** screen
- "ShortLO" – is the input which we have to provide in the SOAP message
- "ShortOPLO" – is the output which we have to provide in the SOAP message
- "errorCode" – is for error messages, so if the functionality is not correct or we have wrong SOAP message the error code will come out in outgoing SOAP messages

The resultant operation segment is generated in three parts:

- soakRequestType
- soakResultType
- soakFaultType

```
<xs:complexType name="soakRequestType">
  <xs:sequence>
    <xs:element name="CC_Service_Number"
type="xmlns.oracle.com/communications/ncc:NumericString" minOccurs="1"/>
    <xs:element name="ShortLO" type="xmlns.oracle.com/communications/ncc:Short" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="soakResultType">
```

```
  <xs:sequence>
    <xs:element name="ShortOPLO" type="xmlns.oracle.com/communications/ncc:Short"
minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="soakFaultType">
  <xs:sequence>
    <xs:element name="errorCode" type="xs:int"/>
  </xs:sequence>
</xs:complexType>
```

Operation WSDL

This is the operation generated WSDL code from the control plan compilation.

```
  <types>
    <xs:element name="soakRequest" type="soakRequestType"/>
    <xs:element name="soakResult" type="soakResultType"/>
    <xs:element name="soakFault" type="soakFaultType"/>
    <xs:complexType name="soakRequestType">
      <xs:sequence>
        <xs:element name="CC_Service_Number"
type="xmlns.oracle.com/communications/ncc:NumericString" minOccurs="1"/>
        <xs:element name="ShortLO" type="xmlns.oracle.com/communications/ncc:Short"
minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="soakResultType">
      <xs:sequence>
        <xs:element name="ShortOPLO" type="xmlns.oracle.com/communications/ncc:Short"
minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="soakFaultType">
      <xs:sequence>
        <xs:element name="errorCode" type="xs:int"/>
      </xs:sequence>
    </xs:complexType>
  </types>

  <message name="soakInput">
    <part name="body" element="tns:soakRequest"/>
  </message>
  <message name="soakOutput">
    <part name="body" element="tns:soakResult"/>
  </message>
  <message name="soakFaultOutput">
    <part name="body" element="tns:soakFault"/>
  </message>

  <portType name="soakPortType">
    <operation name="soakOperation">
      <input message="tns:soakInput"/>
      <output message="tns:soakOutput"/>
      <fault message="tns:soakFaultOutput"/>
    </operation>
  </portType>

  <binding name="soakBinding" type="tns:soakPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="soakOperation">
      <soap:operation soapAction="http://boss-sb-smp/wsdls/OSD/soak/soak"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
      <fault>
        <soap:body use="literal"/>
```

```
      </fault>
    </operation>
  </binding>
```

Operation Set WSDL

This is the full WSDL generated for the operation set.

Note: The "===" lines are not part of the generated file,they mark the start and end lines of the operation WSDL lines.

```
<?xml version="1.0"?>
<definitions name="soak"
  targetNamespace="http://boss-sb-smp/wsdls/OSD/soak.wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://boss-sb-smp/wsdls/OSD/soak.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
============
  <types>
    <xs:element name="soakRequest" type="soakRequestType"/>
    <xs:element name="soakResult" type="soakResultType"/>
    <xs:element name="soakFault" type="soakFaultType"/>
    <xs:complexType name="soakRequestType">
      <xs:sequence>
        <xs:element name="CC_Service_Number"
type="xmlns.oracle.com/communications/ncc:NumericString" minOccurs="1"/>
        <xs:element name="ShortLO" type="xmlns.oracle.com/communications/ncc:Short"
minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="soakResultType">
      <xs:sequence>
        <xs:element name="ShortOPLO" type="xmlns.oracle.com/communications/ncc:Short"
minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="soakFaultType">
      <xs:sequence>
        <xs:element name="errorCode" type="xs:int"/>
      </xs:sequence>
    </xs:complexType>
  </types>

  <message name="soakInput">
    <part name="body" element="tns:soakRequest"/>
  </message>
  <message name="soakOutput">
    <part name="body" element="tns:soakResult"/>
  </message>
  <message name="soakFaultOutput">
    <part name="body" element="tns:soakFault"/>
  </message>

  <portType name="soakPortType">
    <operation name="soakOperation">
      <input message="tns:soakInput"/>
      <output message="tns:soakOutput"/>
      <fault message="tns:soakFaultOutput"/>
    </operation>
  </portType>

  <binding name="soakBinding" type="tns:soakPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="soakOperation">
      <soap:operation soapAction="http://boss-sb-smp/wsdls/OSD/soak/soak"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
```

```
      </output>
      <fault>
        <soap:body use="literal"/>
      </fault>
    </operation>
  </binding>
============
  <service name="soak">
    <port name="soakPort1" binding="tns:soakBinding">
      <soap:address location="http://eng-host06-z7:6262"/>
    </port>
  </service>
</definitions>
```

Incoming SOAP message

This is the incoming SOAP message used to trigger the control plan.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tags="http://abox-sb-smp/wsdls/OSD/tags.wsdl">
<soapenv:Header/>
  <soapenv:Body>
    <tags:acsTagsRequest xmlns="http://abox-sb-smp/wsdls/OSD/tags.wsdl">
      <!--Optional:-->
      <Short>12</Short>
      <CC_Service_Number>1234567</CC_Service_Number>
    </tags:acsTagsRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Outgoing SOAP message

This is the outgoing SOAP message.

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">
  <SOAP-ENV:Body>
    <m:reTestResult xmlns:m="http://abox-sb-smp/wsdls/OSD/reTest.wsdl">
      <ShortOPLO>12</ShortOPLO>
    </m:reTestResult>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Service Handlers

Introduction

Service handlers determine which service is used to invoke a control plan from an OSD WSDL file. Each service handler must match a configured service on the SLC. You associate a service handler with an OSD operation by selecting the service handler from the **Service to Invoke** list for an operation set in the OSD UI. The list of available services depends on which NCC components are installed on the platform. See *Operations* (on page 22) for more information.

For meanings and uses of the different service handlers, see *ACS Technical Guide*, *CCS Technical Guide*, and *Notification Gateway Technical Guide.*

For information about the mandatory parameters required by the different services, see *Mandatory Parameters for OSD* (on page 45).

OSD service handlers

The following list shows the possible services that you can invoke when you install OSD.

- ACS
- ACS_Management
- ACS_OSD
- ACS_Outgoing
- ACS_Prefix

CCS service handlers

The following list shows the possible Prepaid Charging services that you can invoke through OSD when you install the CCS:

- CCS
- CCS_ROAM
- CCS_SM_MO
- CCS_SM_MT
- REVERSE_CCS_SM_MT
- CCS_DATA
- CCS_MO
- CCS_MT
- CCS_BPL

NGW Ext_Sub service handler

The Notification Gateway uses the Ext_Sub service handler. Ext_Sub handles notifications for external subscribers that are received from the Notification Gateway. The Ext_Sub service maps external balances, such as ECE subscriber balances, to NCC balances. The NCC balances are then used in notification templates referenced by OSD control plans.

See *Notification Gateway Technical Guide* for information about Ext_Sub service handler configuration.

Chapter 2

Getting Started

Overview

Introduction

This chapter explains how to start the Open Services Development (OSD) user interface (UI).

In this chapter

This chapter contains the following topics.

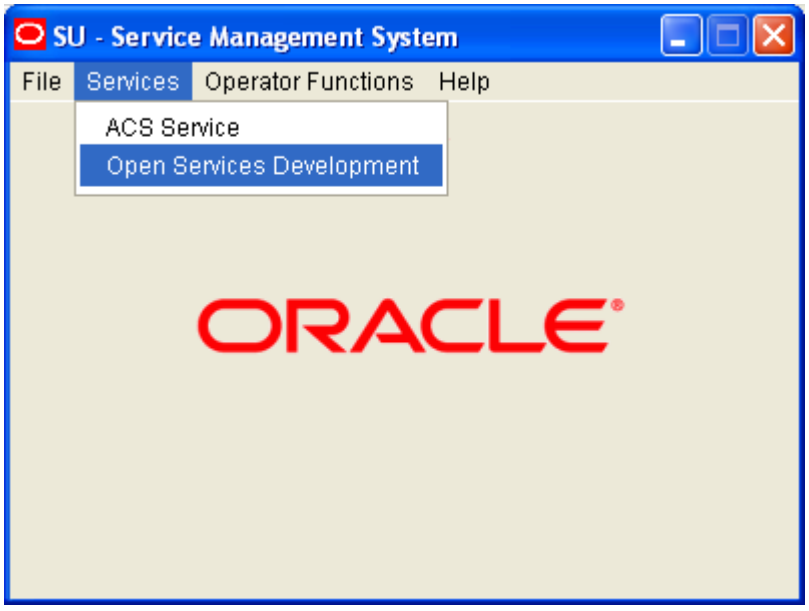
Signing on to Open Services Development

Overview

You access Open Services Development (OSD) by logging into the Service Management System (SMS) and selecting Open Services Development from the Services menu in the Service Management System window. For more information about the SMS user interface, see *Service Management System User's Guide*.

SMS main screen

Here is an example of the Service Management System main menu showing the Open Services Development menu option.



Chapter 3

Using Open Services Development Screens

Overview

Introduction

The Open Services Development (OSD) user interface (UI) enables you to configure how incoming SOAP messages will be handled for any OSD data that is specific to a service provider, that is, one ACS customer. This chapter explains how to configure OSD in the OSD UI.

In this chapter

This chapter contains the following topics.

Find Screens

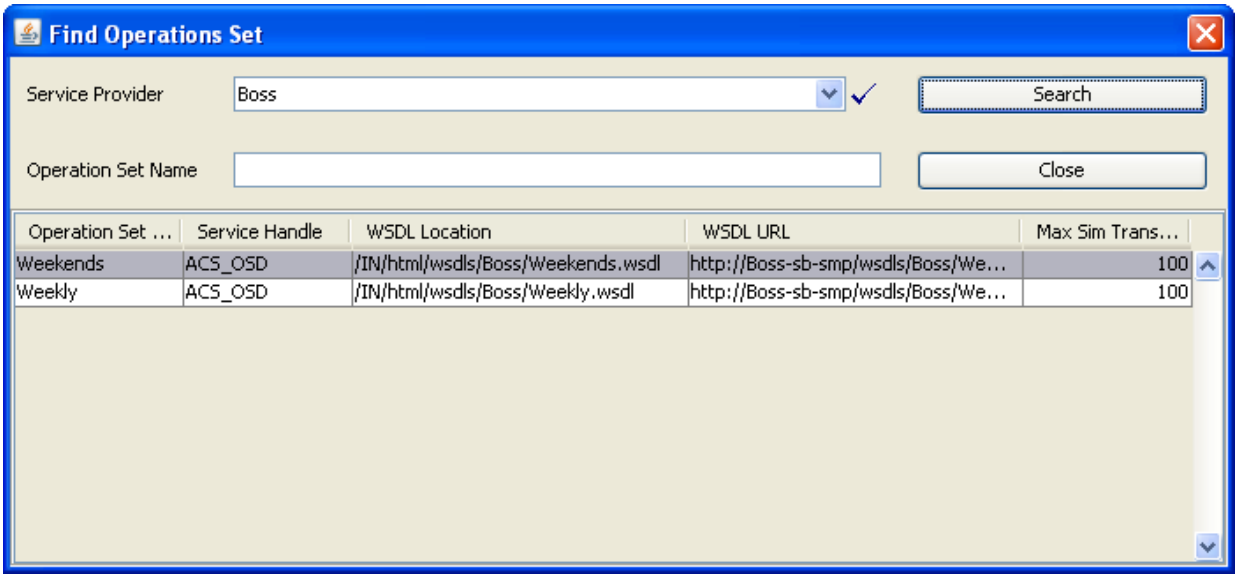
Introduction

You use the Find screens to locate records in the **Operation Sets**, **Operations** and **Client ASPs** tabs.

While each of these tabs has a different results table, they all use the same mechanism to populate their tables.

Example Find screen

Here is an example find screen with the default results table for the selected service provider.



Using the Find screen

If the **Service Provider** drop down list is present, it can be ignored, unless you are wanting to change provider on all the tabs.

The search text box (**Operation Set Name** in the example) is used to do a search on items beginning with the text typed. For example typing "Weekly" would return all operation sets whose name started with "Weekly".

Note: This search function is case sensitive - "week" would not find anything.

To start the search, click **Search**. All found items replace any previous table contents.

Once the record has been found, click on that table entry and click **Close** to return to the parent tab, which will be populated with the found record details.

Service Providers

Introduction

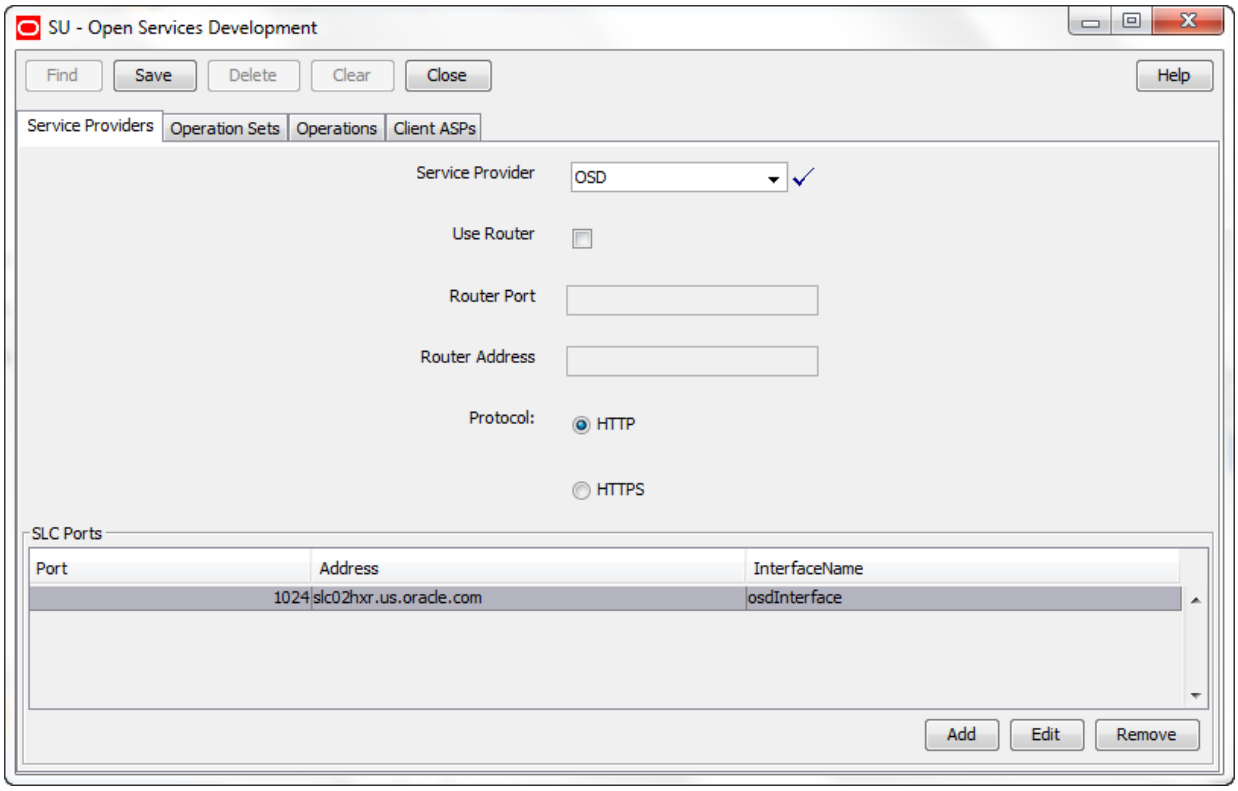
The **Service Providers** tab allows you to select a service provider that will have OSD configuration.

Note: Once selected, data for the service provider is propagated to the following tabs:

- Service Providers
- Operation Sets
- Operations

Service Providers tab

Here is an example of the **Service Providers** tab.



Service Providers fields

This table describes the **Service Providers** tab fields.

Field	Description
Service Provider	The service provider that an ASP uses for the operation set. **Note:** • This will be the same provider for the Service Provider, Operation Sets and Operation tabs. • Selection of a different provider changes the screen contents as if **Clear** had been clicked.
Use Router	Flag to indicate that ASPs using operations belonging to this service provider access OSD through a router such as squid. If this flag is selected then the router port and router address are placed in the WSDL file. If this flag is not selected then ASPs access OSD on the SLCs directly for these operations. In this case, all the ports and addresses from the SLC ports panel are placed in the WSDL file and the router port and router address are not used.
Router Port	This is a single port in the range 1024 through 65535.
Router Address	Address of the router to use for load sharing.
Protocol	The protocol the ASPs should use to send the SOAP request envelope.

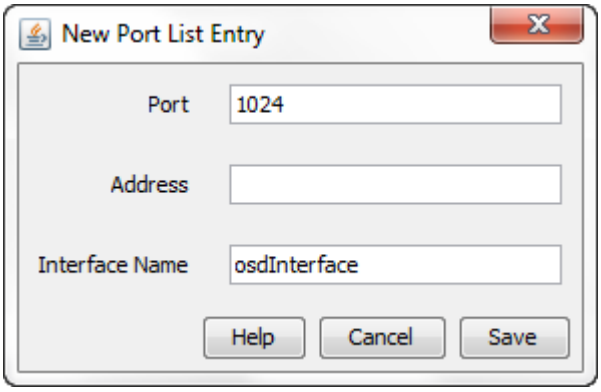
Edit Service Providers

Follow these steps to edit a service provider OSD interaction.

Step	Action
1	Select the **Service Provider** from the drop down list. **Note:** • This is a list of already established service providers (see **SMS Main menu > Services > ACS Services > Customers** tab. • The selected service provider is auto selected in the other tabs.
2	If load sharing is required, select **Use Router** check box, then: • Enter the router port in the **Router Port** field • Enter the router address in the **Router Address** field
3	Select the Protocol to be used: • HTTP • HTTPS
4	Amend the list of SLC ports to receive ASP input from. • To add a new port, see *Adding SLC ports* (on page 18) • To change a port, see *Editing SLC ports* (on page 19) • To delete a port, select the port in the table and click **Remove**
5	Click **Save**.

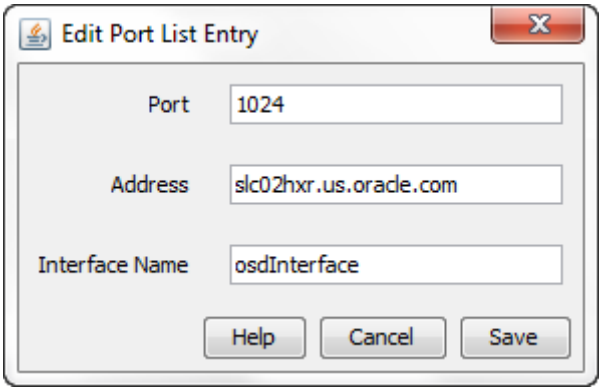
Adding SLC ports

Follow these steps to add a SLC port.

Step	Action
1	Click **Add**. The New Port List Entry screen displays. 
2	Enter the new port number in the **Port** field. You must enter a value in the range 1024 to 65535.
3	Enter the port address in the **Address** field. You must enter the host name of a SLC running OSD.
4	Enter the interface name in the **Interface Name** field. You must enter a name that matches the name of a running osdInterface on the SLC, as defined in the **SLEE.cfg** file. To help improve performance, configure ports for more than one interface.
5	Click **Save**.

Editing SLC ports

Follow these steps to edit an existing SLC port list entry.

Step	Action
1	Select the port entry that you want to change from the table.
2	Click **Edit**. The Edit Port List Entry screen displays. 
3	(Optional) Enter a different port address in the **Address** field. You must specify the host name of a SLC running OSD.
4	(Optional) Type a different interface name in the **Interface Name** field. You must enter the name of a running osdInterface on the SLC, as defined in the **SLEE.cfg** file.
5	Click **Save**.

Operation Sets

Introduction

Operation sets are a collection, for ease of maintainability, of related operations.

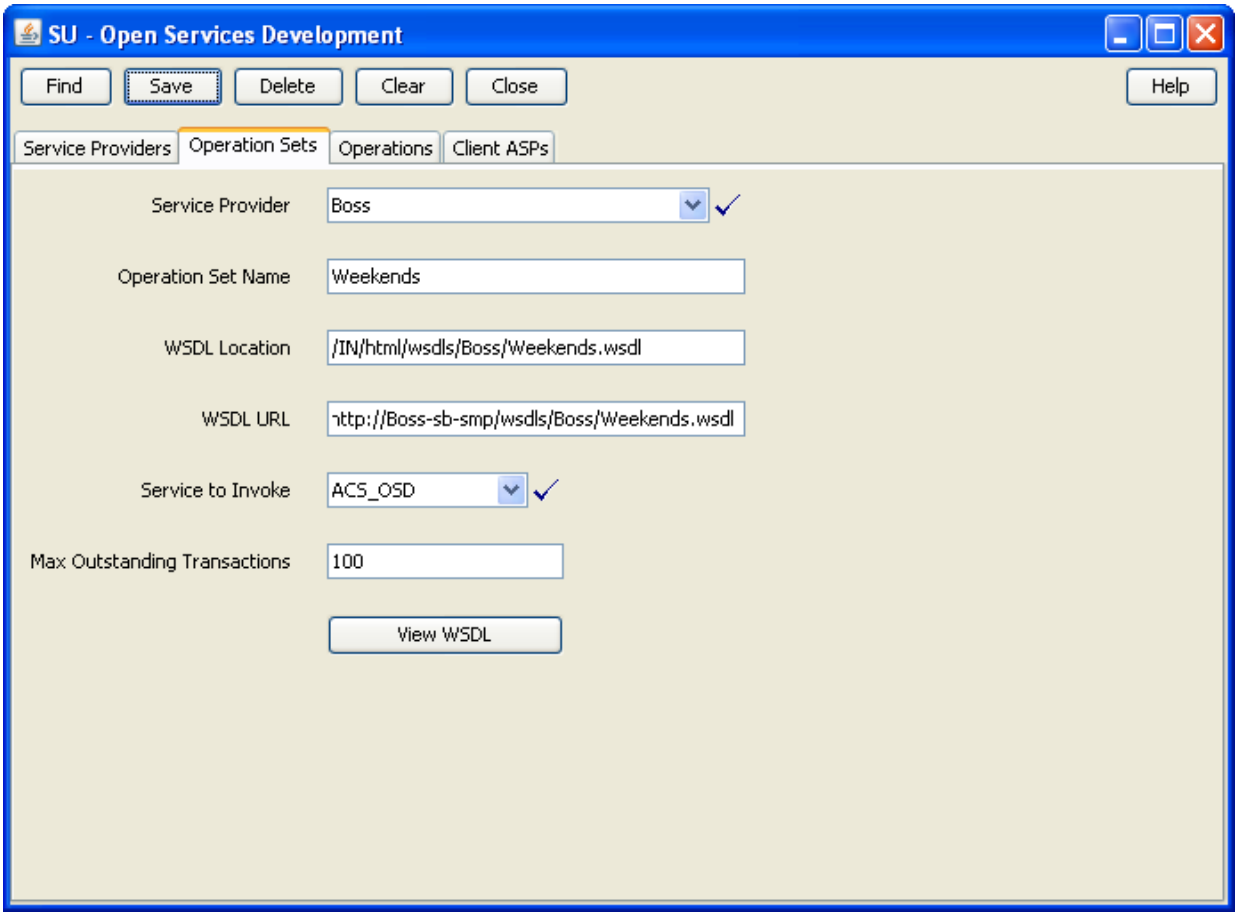
The **Operation Sets** tab is where the selected service provider has all their sets of operations configured.

Each set can have any number of operations (see *Operations* (on page 22)) and each service provider can have any number of operation sets.

When generated by a control plan compile, all operations for the operation set are inserted into a single WSDL file.

Operation Sets tab

Here is an example of the **Operation Sets** tab.



Operation Sets fields

This table describes the **Operation Sets** tab fields.

Field	Description
Service Provider	The service provider for this operation set. **Note:** • This will be the same provider for the Service Provider, Operation Sets and Operation tabs. • Selection of a different provider changes the screen contents as if **Clear** had been clicked.
Operation Set Name	The name of this collection of operations. **Note:** There is a special PeriodicCharge set name for use with period charging pro-rating.
WSDL Location	Location of the control plan generated WSDL file for this operation set. **Note:** First part of this is set at installation time in the **smsGui.bat/smsGui.sh** file.
WSDL URL	Web URL for the WSDL file.

Field	Description
	Note: This is set at installation time in the **smsGui.bat/smsGui.sh** file and in **eserv.config** - `wsdlUriBaseName` parameter.
Service to Invoke	The service that will be used to invoke the control plan from the WSDL file. This must match a configured service on the SLC to trigger the control plan successfully. **Note:** This list is created at SMS package installation time in a database table. See Service Handlers. For meanings and uses of the different service handlers, see *ACS Technical Guide*, *CCS Technical Guide*, and *Notification Gateway Technical Guide*.
Max outstanding Transactions	This is the maximum number of SOAP requests for operations in this operation set that are allowed to be active at any one time. Any SOAP requests in excess of this will be rejected with HTTP error 503 (unavailable).

Warning: If either the WSDL location or WSDl URL are modified, then it is up to the user to configure the SMS file system and Apache so that the two are consistent.

Editing Operation Sets

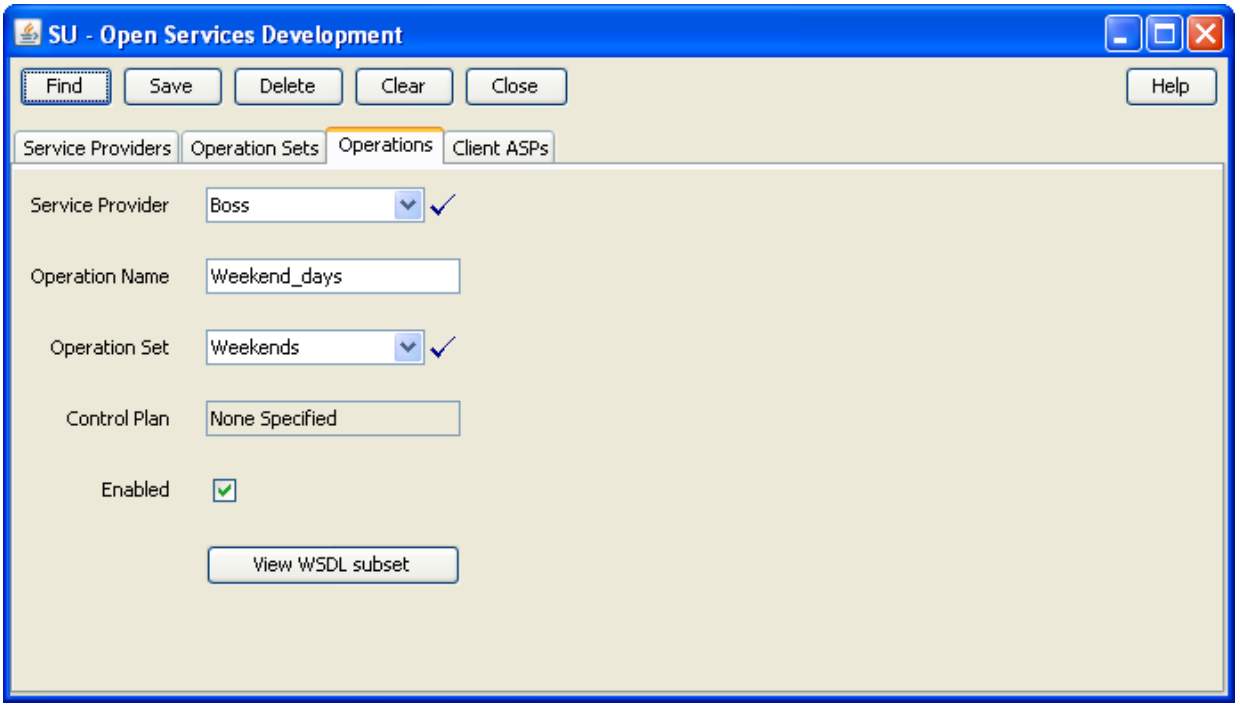
Follow these steps to edit a service provider operation set.

Step	Action
1	Select the **Service Provider** from the drop down list. **Note:** • This is a list of already established service providers (see **SMS Main menu > Services > ACS Services > Customers** tab. • The selected service provider is auto selected in the other tabs.
2	To set the tab for a new operation set, click **Clear**. To locate an existing operation set for amending, click **Find** (see *Find screens* (on page 15)). To remove an operation set, click **Delete**, and then confirm on the Delete Operation Set confirmation dialog.
3	Type the operation set name in the **Operation Set Name** field. **Result:** The name is inserted into the WSDL file location and URL.
4	Select the **Service to Invoke** from the drop down list. **Tip:** This should reflect the service the WSDL is going to invoke.
5	Type the maximum allowed outstanding transactions in the **Max Outstanding Transactions** field.
6	Click **Save**.

Operations

Operations tab

Here is an example of the **Operations** tab.



Operations fields

This table describes the **Operations** tab fields.

Field	Description
Service Provider	The service provider for for this operation. **Note:** • This will be the same provider for the **Service Provider**, **Operation Sets** and **Operation** tabs. • Selection of a different provider changes the screen contents as if **Clear** had been clicked.
Operation Name	The name of this operation. This is the name that the control plan uses when generating the WSDL file sub set.
Operation Set	The operation set that this operation will belong to. **Tip:** For periodic charges this must be `PeriodicCharge`.
Control Plan	The control plan that this WSDL sub set will invoke. This is automatically populated when saving a control plan with this operation name.
Enabled	If an operation is not enabled, the ASP will receive a SOAP fault with error 7 = operation not available.

Editing Operations

Follow these steps to edit a service provider operation.

Step	Action
1	Select the **Service Provider** from the drop down list.
2	To set the tab for a new operation, click **Clear**. To locate an existing operation for amending, click **Find** (see *Find screens* (on page 15)). To remove an operation, click **Delete**, then confirm on the Delete confirmation dialog.
3	Type the operation name in the **Operation Name** field. **Result:** The name is inserted into the WSDL file location and URL.
4	Select the **Operation Set** for this operation from the drop down list.
5	Select the **Enabled** check box.
6	Click **Save**.
7	Repeat steps **4** to **6** for each operation set that you want to add this operation to.

Updating Operation Set or Operation Name in Operations Tab

When updating the Operation Name or Operation Set in the Operations tab in DAP, it is necessary to regenerate both the WSDL files associated with the operation. This ensures that the changes are correctly reflected and that no outdated or duplicate information remains.

Follow these steps to update the operation sets.

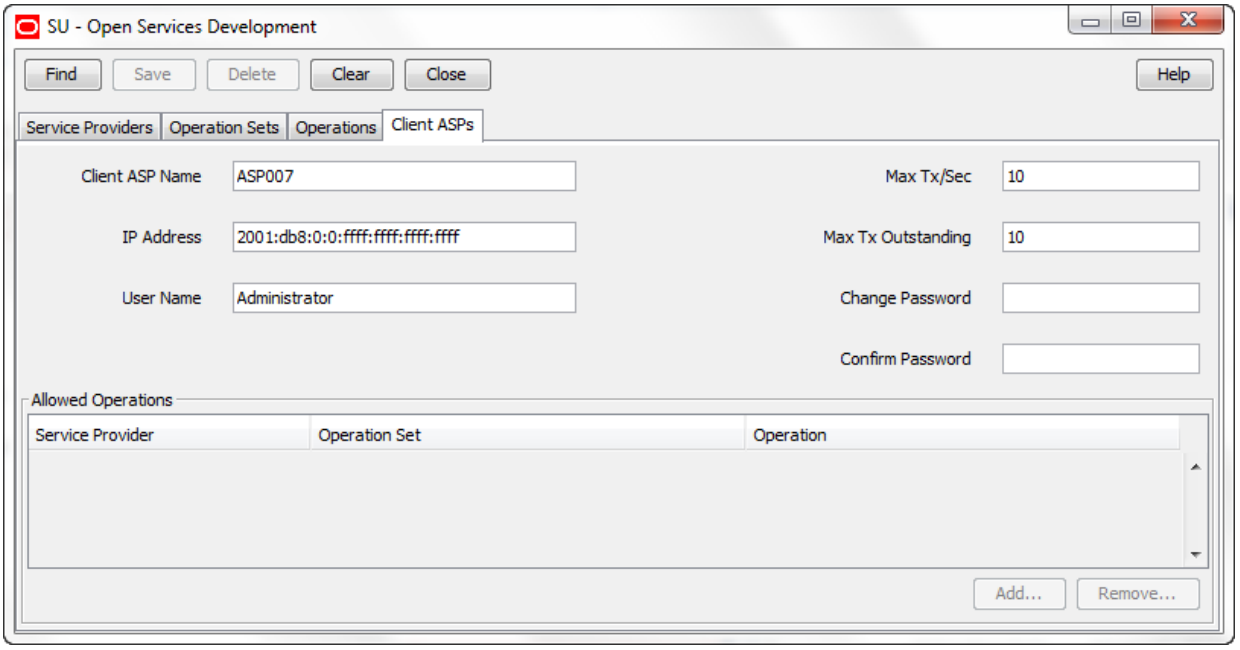
Step	Action
1	Access the Operations screen. Navigate to Services > DAP > Resources > Operations tab.
2	Update the Operation Set: a) Select the desired operation. b) Change the Operation Set value from the dropdown menu. c) Click the save button to apply the changes.
3	Regenerate WSDL Files: a) Identify the WSDLs: Determine the WSDL file previously linked to the operation and the new WSDL file linked after the update. b) Regenerate from Control Plan: Use the control plan linked with the particular operation to regenerate both WSDL files. c) Ensure both the old and new WSDLs reflect the updated Operation Name information.
4	Verification: a) Verify that the operation is listed correctly in both WSDL files. b) Ensure there are no duplicate entries or outdated information. c) Once both WSDLs are regenerated, the WSDL operation should show the correct control plan name when triggered.

Note: Follow the above steps for updating Operation Name as well.

Client ASPs

Client ASPs tab

Here is an example **Client ASPs** tab.



Client ASPs fields

This table describes the function of each field.

Field	Description
Client ASP Name	The name of the ASP.
IP Address	The IP address for the ASP. IP class is also supported. For example, 198.51.100.250/24.
User Name	The user name for this ASP. The combination of user name and IP address is used to identify the ASP.
Max Tx/Sec	The maximum number of SOAP requests per second this ASP is allowed to send. Any SOAP requests in excess of this will be rejected with HTTP error 503 (unavailable).
Max Tx Outstanding	The maximum number of SOAP requests from this ASP that are allowed to be active at any one time. Any SOAP requests in excess of this will be rejected with HTTP error 503 (unavailable).
Change Password	The password used to authenticate this ASP.
Confirm Password	This must match Change Password value.

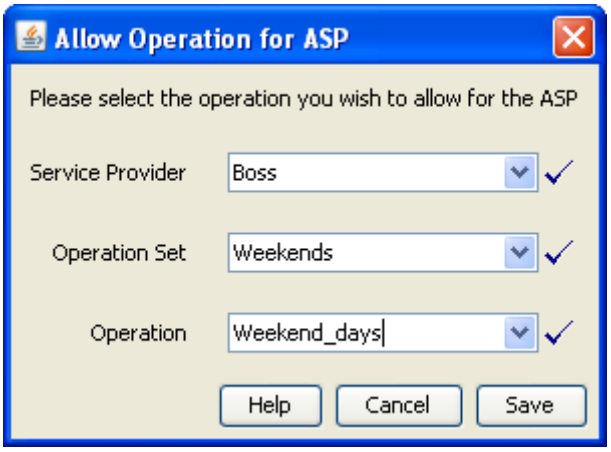
Edit Client ASPs

Follow these steps to edit a client ASP.

Step	Action
1	To amend or delete an ASP, use the **Find** functionality to locate the required ASP (see *Find screens* (on page 15)).
2	To add a new ASP, click **Clear**.
3	Enter the new ASP name in the **Client ASP Name** field.
4	Enter or change the **IP Address**.
5	Enter or change the **User Name**.
6	Enter or change the maximum transaction rate in the **Max Tx/Sec** field.
7	Enter or change the maximum transaction backlog in the **Max Tx Outstanding** field.
8	Set the password for the SOAP HTML header in the **Change Password** and **Confirm Password** fields. **Note:** Any password will do, but a secure password containing characters and numbers is recommended.
9	Click **Save**. **Result:** The **Add..** becomes available to do the next step.
10	Maintain the allowed operations for this ASP (see *Add Allowed Operations* (on page 25)).

Add Allowed Operations

Follow these steps to add an allowed operation.

Step	Action
1	Click **Add**. **Result:** The Allow Operation for ASP screen appears, with the fields defaulted as follows: • Service provider is the currently selected provider in other tabs • Operation set is the first in the list for the provider • Operation is the first in the list for the operation set 
2	If required, select a new **Service Provider** from the drop down list. **Note:** The selected provider is also changed in the other OSD tabs and a prompt is made if there are unsaved changes.

Step	Action
3	If required, select the **Operation Set** from the drop down list.
4	If required, select the **Operation** from the drop down list.
5	Click **Save**. **Result:** The operation is added to the **Allowed Operations** table.

Remove Allowed Operations

Follow these steps to remove an allowed operation from this ASP.

Step	Action
1	Select the allowed operation to remove from this ASP from the **Allowed Operations** table.
2	Click **Remove...**. **Result:** The confirmation screen appears.
3	Click **Remove** to confirm the removal. **Result:** The allowed operation is removed from the list.

Notification Gateway User

Introduction

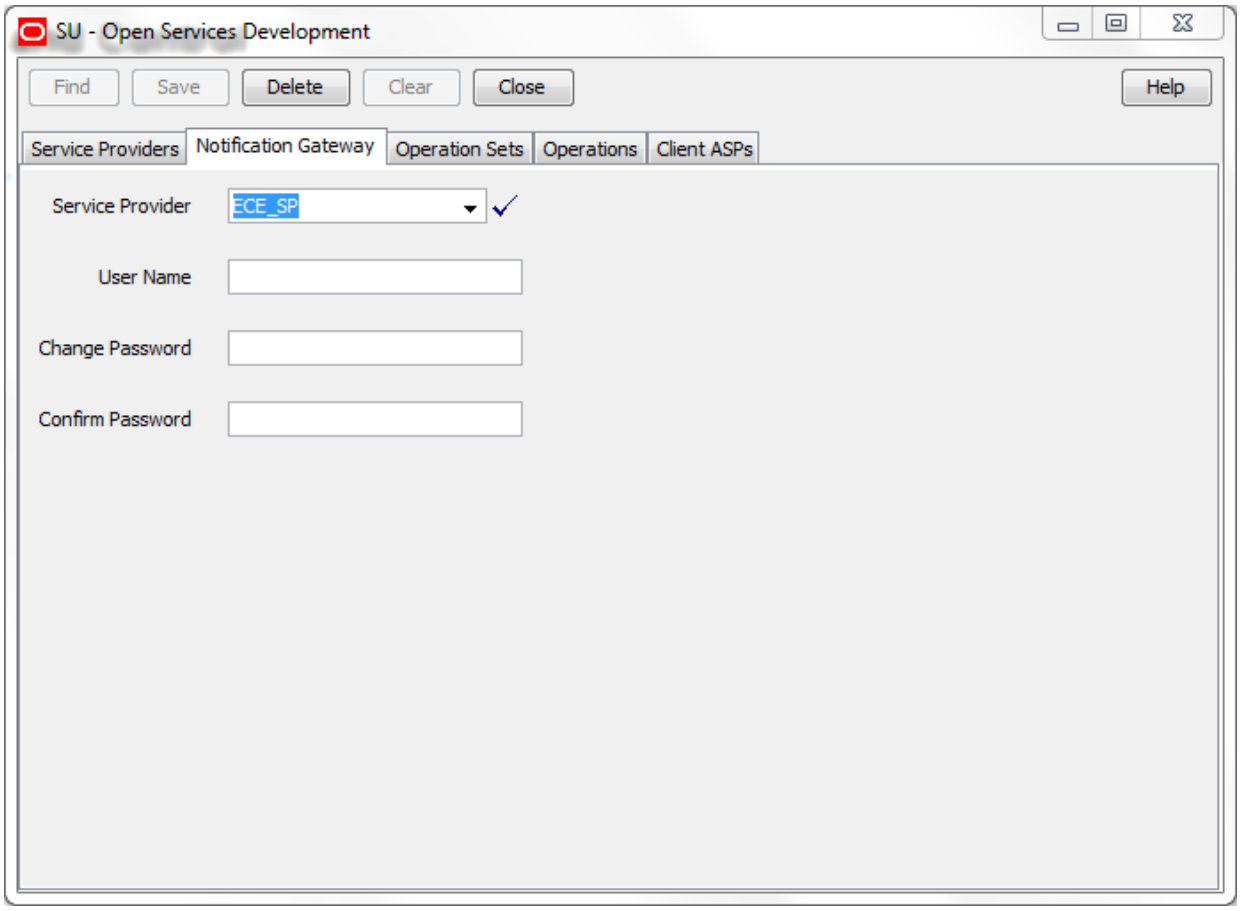
The notification gateway user enables the Notification Gateway to access OSD remotely. You set the user credentials (username and password) for the notification gateway user on a service provider basis, on the **Notification Gateway** tab in the OSD UI. The username and password are stored in a secure credentials vault on the SMS.

The **Notification Gateway** tab is available in the UI only if the `jnlp.ECEExtensions` Java application property is present and set to `true` in the **/IN/html/smsGui.bat** and **/IN/html /smsGui.sh** configuration file. See *smsGui script Configuration* (on page 43) for more information.

Note: You can override user credentials by setting the [SERVICE/USER] and [SERVICE/PASS] parameters in the Notification Gateway **config.xml** file. You should set these parameters only if you do not want to store user credentials in the NCC secure credentials vault. See *Notification Gateway Technical Guide* for more information.

Notification Gateway tab

Here is an example **Notification Gateway** tab.



Setting the Notification Gateway Username and Password

Follow these steps to set the user credentials for the notification gateway user for a selected service provider.

Step	Action
1	Select the **Notification Gateway** tab in the Open Services Development window.
2	Select the **Service Provider** from the drop down list.
3	Enter the name of the authorized user of the Notification Gateway in the **User Name** field.
4	Enter a new password for the user in the **Change Password** field.
5	Re-enter the password in the **Confirm Password** field.
6	Click **Save**. **Result:** The user credentials (username and password) are stored in the NCC secure credentials vault on the SMS.

Chapter 4

Configuration

Overview

Introduction

This chapter explains how to configure the Oracle Communications Network Charging and Control (NCC) application.

In this chapter

This chapter contains the following topics.

eserv.config Configuration

Introduction

The **eserv.config** file is a shared configuration file, from which many Oracle Communications Network Charging and Control (NCC) applications read their configuration. Each NCC machine (SMS, SLC, and VWS) has its own version of this configuration file, containing configuration relevant to that machine. The **eserv.config** file contains different sections; each application reads the sections of the file that contains data relevant to it.

The **eserv.config** file is located in the **/IN/service_packages/** directory.

The **eserv.config** file format uses hierarchical groupings, and most applications make use of this to divide the options into logical groupings.

Configuration File Format

To organize the configuration data within the **eserv.config** file, some sections are nested within other sections. Configuration details are opened and closed using either { } or [ ].

- Groups of parameters are enclosed with curly brackets – { }
- An array of parameters is enclosed in square brackets – [ ]
- Comments are prefaced with a # at the beginning of the line

To list things within a group or an array, elements must be separated by at least one comma or at least one line break. Any of the following formats can be used, as in this example:

```
{ name="route6", id = 3, prefixes = [ "00000148", "0000473"] }
{ name="route7", id = 4, prefixes = [ "000001049" ] }
```

or

```
{ name="route6"
    id = 3
    prefixes = [
        "00000148"
```

```
            "0000473"
        ]
    }
    { name="route7"
        id = 4
        prefixes = [
            "000001049"
        ]
    }
```

or

```
    { name="route6"
        id = 3
        prefixes = [ "00000148", "0000473" ]
    }
    { name="route7", id = 4
        prefixes = [ "000001049" ]
    }
```

eserv.config Files Delivered

Most applications come with an example **eserv.config** configuration in a file called **eserv.config.example** in the root of the application directory, for example, **/IN/service_packages/eserv.config.example**.

Editing the File

Open the configuration file on your system using a standard text editor. Do not use text editors, such as Microsoft Word, that attach control characters. These can be, for example, Microsoft DOS or Windows line termination characters (for example, ^M), which are not visible to the user, at the end of each row. This causes file errors when the application tries to read the configuration file.

Always keep a backup of your file before making any changes to it. This ensures you have a working copy to which you can return.

Loading eserv.config Changes

If you change the configuration file, you must restart the appropriate parts of the service to enable the new options to take effect.

Re-reading the config file

The system will re-read config on a SIGHUP signal and print a status report to standard output on a SIGUSR1 signal.

OSD eserv.config configuration

Here is an example OSD **eserv.config** section configuration.

```
OSD = {

    osdInterface = {
        allowINSECURESSLv3 = false
        basicRealm = "ASP Account"
        pollTime = 10000
        loadReportingPeriod = 600
        oracleusername = "smf"
        oraclepassword = "smf"
        oracledatabase = "nzwn-test08_SMF"
        overrideWsdlNamespaceAliasErrorCondition = false
        sslCertificateFile = "/IN/service_packages/OSD/sslCertificate.pem"
        sslKeyFile = "/IN/service_packages/OSD/sslKey.pem"
        tssfTimeOutSecs = 10
```

```
        persistentConnection = true
        connectionTimeOutSecs = 60
        rateCalculationPeriodSecs = 10
        applicationContext = "0,4,0,0,1,21,3,4"
        validateRequestNameSpace = false
        databaseCachingRules = {
            operationSetsDataExpirySecs = 300
            operationDataExpirySecs = 300
            clientAspDataExpirySecs = 300
            portListsDataExpirySecs = 300
            acsProfileDataExpirySecs = 300
            mandatoryParameterDataExpirySecs = 300
        }
        NumberRules = [
            { prefix="", min=0, max = 100, remove=0, prepend="", resultNoa=4 }
        ]
    }

    wsdlUriBaseName = "http://nzwn-test08.uk.oracle.com/wsdls"
    useDeprecatedSchema = false
    useHostnameAndPort = false
    maxProfileDetailsAge = 60
    maxServiceHandleAge = 60
    osdWsdlRegenerator = {
        waitTimeSecs = 2
        useOracleAlerts = false
        oracledatabase = "/"
    }
    osdMacroNodes = {
              skipEmptyChild = true
        }
}
```

OSD SLC parameters

Here are the parameters in the `osdInterface` section of the **eserv.config** OSD configuration.

Note: `osdInterface` parameters are only relevant on an SLC.

`allowINSECURESSLv3`

Syntax: `allowINSECURESSLv3 = true|false`

Description: Whether to allow use of SSLv3 in the SSL handshake for SSL enabled systems. For example, set this parameter to true for customers with an ASP that must use the SSLv3 protocol version. Use of SSLv3 and SSLv2 is disabled by default.

Type: Boolean

Optionality: Optional (default used if not set)

Allowed:

- true – Use of SSLv3 protocol version enabled.
- false – Use of SSLv3 protocol version disabled.

Default: false

Notes: The `allowINSECURESSLv3` parameter can be set for the DAP, PI and OSD components. You should set `allowINSECURESSLv3` to true if the ASP is able to use only SSLv3 protocol version. Otherwise set `allowINSECURESSLv3` to false.

Example: `allowINSECURESSLv3 = true`

applicationContext

Syntax: `applicationContext = "context"`
Description: The application context to specify in IDPs sent to slee_acs.
Type: String
Optionality: Optional (default used if not set).
Allowed:
Default: "0,4,0,0,1,21,3,4"
Notes: This should not normally be changed.
Example: `applicationContext = "0,4,0,0,1,21,3,4"`

basicRealm

Syntax: `basicRealm = "realm"`
Description: The basic realm to specify in HTTP authentication (401) messages, WWW-Authenticate header.
Type: String
Optionality: Optional (default used if not set).
Allowed:
Default: "ASP Account"
Notes:
Example: `basicRealm = "ASP Account"`

connectionTimeOutSecs

Syntax: `connectionTimeOutSecs = seconds`
Description: The http connection timeout in seconds.
Type: Integer, Decimal, Array, Parameter list, String, Boolean
Optionality: Optional (default used if not set).
Allowed:
Default:
- 60 for persistent connection mode
- 10 for single connection mode

Notes: Default depends upon `persistentConnection` setting.
Example: `connectionTimeOutSecs = 60`

loadReportingPeriod

Syntax: `loadReportingPeriod = seconds`
Description: Number of seconds between automatic status reports.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 600
Notes:
Example: `loadReportingPeriod = 900`

oracledatabase

Syntax: `oracledatabase = "database_name"`
Description: The remote Oracle database to connect to, from the **tnsnames.ora** file.

Type: String
Optionality: Optional (default used if not set)
Allowed:
Default: None
Notes: Usually not specified as it is usual to use the local database.
Example: `oracledatabase = "test_SMF"`

Syntax: `oraclepassword = "password"`
Description: Oracle password for connecting to the database.
Type: String
Optionality: Optional
Allowed:
Default:
Notes: Usually the `oracleusername` and `oraclepassword` are not specified in which case the acs_oper operator account will just connect as "/".
Example: `oraclepassword = "smf"`

oracleusername

Syntax: `oracleusername = "user_name"`
Description: Oracle user name for connecting to the database.
Type: String
Optionality: Optional.
Allowed:
Default:
Notes: Usually the `oracleusername` and `oraclepassword` are not specified in which case the acs_oper operator account will just connect as "/".
Example: `oracleusername = "smf"`

osdInterface

Syntax: `osdInterface = {parm1, parm2...}`
Description: The OSD interface parameter list.
Type: List
Optionality: Mandatory
Allowed:
Default: None
Notes: `osdInterface` parameters are only relevant on an SLC.
Example:

```
osdInterface = {
    parm 1
    parm 2
}
```

overrideWsdlNamespaceAliasErrorCondition

Syntax: `overrideWsdlNamespaceAliasErrorCondition = true | false`
Description: Specifies whether error codes include the WSDL namespace prefix.
Type: Boolean
Optionality: Optional (default used if not set)

Allowed:
- true – Error codes do not include the WSDL namespace prefix..
- false – Error codes include the WSDL namespace prefix.

Default: false

Notes:

Example: `overrideWsdlNamespaceAliasErrorCondition = true`

persistentConnection

Syntax: `persistentConnection = true|false`

Description: Run in persistent http connection mode.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed:
- true - run in persistent http connection mode
- false - run in single http connection mode

Default: true

Notes:

Example: `persistentConnection = true`

pollTime

Syntax: `pollTime = microseconds`

Description: Number of microseconds to wait for a socket connection or SOAP request.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 10000

Notes:

Example: `pollTime = 20000`

rateCalculationPeriodSecs

Syntax: `rateCalculationPeriodSecs = seconds`

Description: The number of seconds for which to store transactions when calculating the transaction rate.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 10

Notes: For example with the default of 10 seconds, and with an allowed rate of 200 per second, new transactions will be rejected if the number of transactions started in the last 10 seconds exceeds 2000.

Example: `rateCalculationPeriodSecs = 20`

skipEmptyChild

Syntax: `skipEmptyChild = true|false`

Description: This parameter decides how Iterator Node will treat the incoming child profile tag values. If it is set to true, then empty child profile tag will be skipped, else it expects a value for all child profile tags and errors out if missing.

Type: Boolean

Optionality: Optional (default used if not set).
Allowed: true, false
Default: false
Notes:
Example:
```
OSD = {
osdMacroNodes = {
          skipEmptyChild = true
    }
}
```

sslCertificateFile

Syntax: `sslCertificateFile = "location/file_name"`
Description: Name and location of the SSL certificate file.
Type: String
Optionality: Optional (default used if not set).
Allowed:
Default: "/IN/service_packages/OSD/sslCertificate.pem"
Notes: The **sslCertificate** file must be created manually in order for SSL to work.
Example:
```
sslCertificateFile =
"/IN/service_packages/OSD/sslCertificate.pem"
```

sslKeyFile

Syntax: `sslKeyFile = "location/name"`
Description: Name and location of the SSL key file for certificate.
Type: String
Optionality: Optional (default used if not set).
Allowed:
Default: "/IN/service_packages/OSD/sslKey.pem"
Notes: The **sslKey** file must be created manually in order for SSL to work.
Example: `sslKeyFile = "/IN/service_packages/OSD/sslKey.pem"`

tssfTimeOutSecs

Syntax: `tssfTimeOutSecs = seconds`
Description: The maximum period osdInterface should wait for a response from slee_acs before giving up and sending a SOAP fault.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 10
Notes:
Example: `tssfTimeOutSecs = 20`

validateRequestNameSpace

Syntax: `validateRequestNameSpace = true|false`
Description: Do not allow incorrect WSDL namespace.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed:
- true - namespace in the request tag of the incoming request will be ignored.
- false - the namespace must match the name space of the operation set.

Default: false

Notes:

Example:
```
validateRequestNameSpace = true
```

databaseCachingRules

Syntax:
```
databaseCachingRules = {rules}
```

Description: The list of table re-read time rules.

Type: List

Optionality: Optional (default used if not set).

Allowed:

Default:

Notes:

Example:
```
databaseCachingRules = {
    operationSetsDataExpirySecs = 10
    operationDataExpirySecs = 10
    clientAspDataExpirySecs = 10
    portListsDataExpirySecs = 10
    acsProfileDataExpirySecs = 10
    mandatoryParameterDataExpirySecs = 10
}
```

Syntax:
```
acsProfileDataExpirySecs = seconds
```

Description: The maximum age data from ACS_PROFILE_DETAILS table can be before it is re-read from the database.

Type:

Optionality: Optional (default used if not set).

Allowed:

Default: 10

Notes: Member of the `databaseCachingRules` (on page 36) section.

Example:
```
acsProfileDataExpirySecs = 300
```

Syntax:
```
clientAspDataExpirySecs = seconds
```

Description: The maximum age data from OSD_CLI_ASP and OSD_CLIENT_ASP_ACCESS tables can be before it is re-read from the database.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 10

Notes: Member of the `databaseCachingRules` (on page 36) section.

Example:
```
clientAspDataExpirySecs = 300
```

operationDataExpirySecs

Syntax: `operationDataExpirySecs = seconds`

Description: The maximum age data from OSD_OPERATION table can be before it is re-read from the database.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 10

Notes: Member of the `databaseCachingRules` (on page 36) section.

Example: `operationDataExpirySecs = 300`

operationSetsDataExpirySecs

Syntax: `operationSetsDataExpirySecs = seconds`

Description: The maximum age data from OSD_OPERATION_SET table can be before it is re-read from the database.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 10

Notes: Member of the `databaseCachingRules` (on page 36) section.

Example: `operationSetsDataExpirySecs = 300`

portListsDataExpirySecs

Syntax: `portListsDataExpirySecs = seconds`

Description: The maximum age data from OSD_PORT_LIST and OSD_PORT_LIST_ENTRY tables can be before it is re-read from the database.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 10

Notes: Member of the `databaseCachingRules` (on page 36) section.

Example: `portListsDataExpirySecs = 300`

mandatoryParameterDataExpirySecs

Syntax: `mandatoryParameterDataExpirySecs = seconds`

Description: The maximum age that data from the OSD_MANDATORY_INPUT_PARAMETER table can be before it is re-read from the database.

Type: Integer

Optionality: Optional (default used if not set)

Allowed:

Default: 10

Notes:

Example: `mandatoryParameterDataExpirySecs = 300`

Syntax: `NumberRules = [{rule_1},{rule_2}...]`

Description: Rules for denormalizing numbers to send to slee_acs in an IDP.

Type: Array of number rules.

Optionality: Optional (default used if not set).

Allowed:

Default:
```
NumberRules = [
    { prefix="", min=0, max = 100, remove=0, prepend="",
    resultNoa=4 }
]
```

Notes: The rules below assume that numbers in the XML start with a country code and should be sent in international format (NOA= 4).

Example:
```
NumberRules = [
    { prefix="", min=0, max = 100, remove=0, prepend="",
    resultNoa=4 }
]
```

max

Syntax: `max = len`

Description: The maximum number length.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 999

Notes: Used in number normalization and rules.

Example: `max = 32`

min

Syntax: `min = len`

Description: The minimum number length.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 0

Notes: Used in number normalization and rules.

Example: `min = 4`

prefix

Syntax: `prefix = "digit"`

Description: This rule is applied to numbers with this prefix.

Type: String

Optionality: Optional

Allowed: One or more decimal digits

Default:

Notes: Used in number normalization and rules.

Example: `prefix = "25"`

`prepend`

Syntax:	`prepend= "digits"`
Description:	Determines the digits that are to be prepended to the number, after stripping any as specified previously.
Type:	String
Optionality:	
Allowed:	
Default:	
Notes:	Used in number normalization and rules.
Example:	`prepend = "1111"`

`remove`

Syntax:	`remove = num`
Description:	Determines the number of digits that are stripped from the beginning of the number.
Type:	Integer
Optionality:	
Allowed:	
Default:	
Notes:	Used in number normalization and rules.
Example:	`remove = 2`

`resultNoa`

Syntax:	`resultNoa = noa`
Description:	Resulting NOA after the normalization.
Type:	
Optionality:	
Allowed:	
Default:	
Notes:	Used in number normalization and rules.
Example:	`resultNoa = 4`

OSD SMS parameters

Here are the parameters in the `OSD` section of the **eserv.config**.

Note: The following parameters are only relevant on an SMS.

```
OSD = {


    wsdlUriBaseName = "http://nzwn-test08.uk.oracle.com/wsdls"
    useDeprecatedSchema = false
    useHostnameAndPort = false
    maxProfileDetailsAge = 60
    maxServiceHandleAge = 60
    osdWsdlRegenerator = {
        waitTimeSecs = 2
        useOracleAlerts = false
        oracledatabase = "/"
```

```
    }
}
```

maxProfileDetailsAge

Syntax: `maxProfileDetailsAge = seconds`

Description: The maximum age data from ACS_PROFILE_DETAILS table can be before it is re-read from the database.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 60

Notes:

Example: `maxProfileDetailsAge = 80`

maxServiceHandleAge

Syntax: maxServiceHandleAge `= seconds`

Description: The maximum age data from OSD_SERVICE_HANDLE and OSD_MANDATORY_INPUT_PARAMETER tables can be before data is re-read from the database.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 60

Notes:

Example: `maxServiceHandleAge = 80`

oracledatabase

Syntax: `oracledatabase = '[user/password][@connect_string]'`

Description: Specifies the connection details for connecting to the Oracle database. To connect to the database on a remote host by using SQLnet, set the `oracledatabase` parameter to the TNS database connection string; for example,

`oracledatabase = 'smf/smf@SMF'`

Type: String

Optionality: Optional (default used if not set)

Allowed: For connections to a local database:

- '*user*/*password*' or '/'

For connections to a remote database:

- '*user*/*password@db_sid*'

For connections to a local or a remote database by using the Oracle wallet secure external password store:

- '/*@connection_string*' where *connection_string* is the alias defined for the user and password credentials in the external password store. This alias can be either a TNS name or a service name from **tnsnames.ora**.

Default: '/'

Notes:

Example: `oracledatabase = 'smf/smf'`

osdWsdlRegenerator

Syntax:
```
osdWsdlRegenerator = {parameter_list}
```

Description: The osdWsdlRegenerator parameters define when a WSDL file that has changed will be compiled and the user credentials for logging on to the Oracle database.

Type: List

Optionality: Mandatory

Allowed:
```
osdWsdlRegenerator = {
    waitTimeSecs = int
    useOracleAlerts = true|false
    oracledatabase = '[user/password][@connect_string]'
}
```

Default: See default values for the specific parameter

Notes: To use passwordless connection to the local database on the SMS node, set the `oracledatabase` parameter to the default value: '/'.

To connect to a database on a remote host by using SQLnet, set the `oracledatabase` parameter to the TNS database connection string; for example,

Example:
```
oracledatabase = 'smf/smf@SMF'
osdWsdlRegenerator = {
    waitTimeSecs = 2
    useOracleAlerts = false
    oracledatabase = '/'
}
```

useDeprecatedSchema

Syntax:
```
useDeprecatedSchema = true|false
```

Description: Set true to force use of deprecated old product namespace and schema.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed: true, false

Default: false

Notes: False means using the xmlns.oracle.com/communications/ncc namespace and schema.

Example:
```
useDeprecatedSchema = false
```

useHostnameAndPort

Syntax:
```
useHostnameAndPort = true|false
```

Description: Set true for using combination of host-name and port in PortName field instead of numbers in WSDL.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed: true, false

Default: false

Notes: False means default behaviour of using port number.

Example:
```
useHostnameAndPort = false
```

`useOracleAlerts`

Syntax:	`useOracleAlerts = true|false`
Description:	Whether to use Oracle alerts.
Type:	Boolean
Optionality:	Mandatory
Allowed:	• true - waiting for an Oracle alert • false - sleep and re-read OSD_CHANGED_OPERATION_SET
Default:	None
Notes:	
Example:	`useOracleAlerts = false`

`waitTimeSecs`

Syntax:	`waitTimeSecs = seconds`
Description:	The number of seconds to wait for an Oracle alert or to sleep, depending on the `useOracleAlerts` setting.
Type:	Integer
Optionality:	Mandatory
Allowed:	
Default:	None
Notes:	
Example:	`waitTimeSecs = 2`

`wsdlUriBaseName`

Syntax:	`wsdlUriBaseName = "location"`
Description:	The URL of the directory containing the **xmlns.oracle.com/communications/ncc.xsd** file.
Type:	String
Optionality:	Required
Allowed:	
Default:	`http://IP_of_SMS/wsdls`
Notes:	This must match the corresponding value for the `WSDLURL` parameter in the **smsGui.bat/smsGui.sh** file.
Example:	`wsdlUriBaseName = "http://nzwn-test08.uk.oracle.com/wsdls"`

High volume configuration

If a large number of XML parameters are expected to be passed between osdInterface and slee_acs then the following configuration should be done:

1. Add a number of large SLEE events to **SLEE.cfg** (to get 500 20K events)
   `MAXEVENTS=500 20480`
2. Add the following to the acsChassis section of **acs.conf**
   `minimumSizeOfConnectSleeEvent 20480`

acs.conf Configuration

osd acs.conf configuration

The control plan compiler requires the plug-in **libwsdlGenerator.so** for generating the WSDL file and attaching the control plan to the OSD operation. This configuration is inserted in to the **acs.conf** file on the SMS during installation.

An example `acsCompilerDaemon` section is shown below.

```
acsCompilerDaemon
 CompilerPlugin libwsdlGenerator.so
 alertTimeout 3
 maxBranches 99
 maxNodes 2000
 maxCompiledKb 256
 compressAtKb 128
 compressLevel 1:
```

See *Advanced Control Services Technical Guide* for more **acs.conf** information.

SLEE.config Configuration

osd SLEE.config configuration

When OSD is installed, the **SLEE.cfg** file has the following line added by osdScp:

```
INTERFACE=osdInterface osdInterface.sh /IN/service_packages/OSD/bin EVENT
```

However, if you wanted to loadshare across several osdInterface processes on the same SLC, then a line per interface is required, for example, loadsharing across three interfaces would require:

```
INTERFACE=osdIf1 osdInterface1.sh /IN/service_packages/OSD/bin EVENT
INTERFACE=osdIf2 osdInterface2.sh /IN/service_packages/OSD/bin EVENT
INTERFACE=osdIf3 osdInterface3.sh /IN/service_packages/OSD/bin EVENT
```

Note: The interface names in the SLC ports table would be then be osdIf1, osdIf2 and osdIf3.

smsGui Script Configuration

Notification Gateway ECEExtension parameter

The **Notification Gateway** tab in the OSD UI enables you to set the user credentials for the notification gateway user and to store those credentials securely in a credentials vault on the SMS node. To enable the **Notification Gateway** tab in the OSD UI, you set the `ECEExtensions` parameter to true in the **smsGui.bat/smsGui.sh** file.

See *Notification Gateway Techncical Guide* for more information about Notification Gateway configuration.

`ECEExtensions`

Syntax:	See example
Description:	Enables the **Notification Gateway** tab in the OSD UI when present and set to true. Otherwise the **Notification Gateway** tab is disabled.
Type:	Boolean
Optionality:	Optional (default used if not set)
Allowed:	true false or not set

Default:	Not set
Notes:	
Example:	`-Djnlp.ECEExtensions=true`

WSDL Parameters

The following two Java application properties are required for WSDL file generation. These parameters are automatically added to the **smsGui.bat/smsGui.sh** file when you install OSD:

- `jnlp.osd.WSDLDirectory`
- `jnlp.osd.WSDLURL`

For more information about application properties in smsGui files, see *Customizing the screens* in *Service Management System Technical Guide*.

`WSDLDirectory`

Syntax:	See example
Description:	This is set on install to this value which forms the first part of the Operation Sets WSDL Location field value.
Type:	String
Optionality:	Required
Allowed:	
Default:	
Notes:	Part of OSD. If this parameter value is changed, the parameter `wsdlUriBaseName` in the **eserv.config** file must also be changed.
Example:	`-Djnlp.osd.WSDLDirectory="/IN/html/wsdls"`

`WSDLURL`

Syntax:	`-Djnlp.osd.WSDLURL="`*`url`*`"`
Description:	This is set to the WSDL URL field value (same as `wsdlUriBaseName` parameter), and has the form of: `http://`*`host_name`*`/wsdls`
Type:	String
Optionality:	Optional (default used if not set).
Allowed:	
Default:	
Notes:	Part of OSD. If this parameter value is changed, the parameter `wsdlUriBaseName` in the **eserv.config** file must also be changed.
Example:	`-Djnlp.osd.WSDLURL="http://nzwn-test08.uk.oracle.com/wsdls"`

Chapter 5

Background Processes

Overview

Introduction

This chapter explains the processes that are started automatically by Service Logic Execution Environment (SLEE).

Note: This chapter also includes some plug-ins to background processes which do not run independently.

In this chapter

This chapter contains the following topics.

osdInterface

About the osdInterface

The Open Services Development (OSD) osdInterface is a SLEE interface that accepts SOAP requests from ASPs across a configurable range of TCP/IP ports, and transforms them into IDPs. The IDPs are sent to slee_acs in SLEE events, and SOAP responses are then sent to the requesting clients based on the result of the control plan processed in response to the IDP.

The osdInterface also supports heartbeat messages, received in the form of PING requests, and sends a suitable response before closing the connection. The PING messages are used to check if the server is on-line and therefore able to process requests.

The osdInterface rejects requests that do not contain the mandatory parameters, listed in the table below, for the service being used. The interface returns a fault with errorCode 3 ("Missing parameter") when it rejects a request.

Mandatory Parameters for OSD

The following table lists the services that require mandatory parameters, and the mandatory parameters for each service. The table is ordered alphabetically by service.

Service	Mandatory Parameters
ACS	CC Service Number
ACS_Management	CC Service Number
ACS_Notification	CC Calling Party Id
ACS_Outgoing	CC Calling Party Id
ACS_Prefix	CC Service Number

Service	Mandatory Parameters
CCS	CC Calling Party Id
CCS_BPL	CC Calling Party Id
CCS_DATA	CC Calling Party Id CC Service Number
CCS_MO	CC Calling Party Id CC Service Number
CCS_MT	CC Calling Party Id CC Service Number
CCS_ROAM	CC Calling Party Id CC Service Number
CCS_SM_MO	CC Calling Party Id CC Service Number
CCS_SM_MT	CC Calling Party Id CC Service Number
Ext_Sub	CC Calling Party Id
REVERSE_CCS_SM_MT	CC Calling Party Id CC Service Number

Cached objects

The osdInterface maintains a cache of configuration objects largely sourced from the database.

Some caches will be augmented with runtime information NOT sourced from the database, but needed to be stored on a per-object basis.

Caching is carried out according to the `databaseCachingRules` configuration rules. See *databaseCachingRule* (on page 36)s.

This table lists the caches.

Cache	Description
ASP	Generally sourced from OSD_CLIENT_ASP and OSD_CLIENT_ASP_ACCESS
OperationSet	Generally sourced from OSD_OPERATION_SET
Operation	Fully sourced from OSD_OPERATION
ProfileTag	Fully sourced from ACS_PROFILE_DETAILS and ACS_TAG_TO_PROFILE_MAPPING
SOAPPort	This will be sourced from OSD_PORT_LIST and OSD_PORT_LIST_ENTRY

Throttling

The OSD application uses the osdInterface internal throttling mechanism to reject new requests and return errors. If osdInterface fails to create a SLEE dialog due to overload, it marks itself as overloaded for the rest of the current second.

When overloaded, osdInterface rejects requests to create a SLEE dialog with the following HTTP error:

```
osdInterface is currently overloaded
```

SOAP Requests

The osdInterface will bind to the address/port of all connections defined, and process SOAP requests on all those that are not quiescing (that is, all those that have not been dropped from the configuration table since osdInterface last loaded them from there).

When a connection is dropped from the configuration it will not be removed from the cache at the next cache expiry, but merely updated so that its isQuiescing variable is TRUE. A quiescing connection will be destroyed (and removed from the cache) only when its number of outstanding requests falls to zero.

The SOAP document is parsed and extracted tags of relevance are passed to osdInterface.

Note: Not all tags encountered in the document need be understood, or used.

Restrictions

The following restrictions apply to this version of the interface:

- osdInterface receives SOAP requests over HTTP 1.1 over HTTP or HTTPS
- A single SOAP request per connection will be accepted
- Although the WSDL files specify xs:dateTime for some date types, timezones other than GMT are not supported by osdInterface, that is, the only valid date time format is:
  CCYY-MM-DDThh:*mm*:*ssZ*

Heartbeat message

The OSD interface supports `ping` requests occasionally sent by third-party systems such as M-POS, and responds appropriately before closing the connection.

An `HTTP 1.1 POST` message with the message body `<PING/>` is interpreted as a heartbeat message. This message will be responded to with an `HTTP 1.1 code 204` response (no content) before closing the connection.

Example ping request:

```
POST http://hostname/ HTTP/1.1
... header fields …
<PING/>
```

Example OSD response:

```
HTTP/1.1 204 No Content
```

WSDL Generating Plug-in

Overview

This plug-in is used to generate WSDL files from control plan compilation.

The plug-in must be configured in **acs.conf** as:

```
CompilerPlugin libwsdlGenerator.so
```

The complier daemon runs each plug-in in the order they appear in the configuration file, so the order may become important when more than this plug-in is available.

Note: `libwsdlGenerator.so` is also used for OSD screen configuration changes to regenerate the WSDL file.

WSDL Regenerator

Overview

The WSDL regenerator is used to re-compile WSDL files after any changes made to port lists, operation sets, or operations through the OSD screens.

The re-compile is performed using `libwsdlGenerator.so`.

Statistics Logged

Overview

Statistics are collected using the smsStats statistics methods at appropriate points in the code.

These sets of statistics are gathered to provide the information necessary for the reports.

OSD statistics

This table lists the statistics collected.

Statistic	Description
ASP_REQUESTS_PROCESSED_SUCCESSFULLY	The number of successful SOAP requests from each Client ASP.
ASP_REQUESTS_RECEIVED	The number of SOAP requests from each Client ASP.
ASP_UNSUCCESSFUL_REQUESTS	The number of unsuccessful SOAP requests from each Client ASP.
TOTAL_REQUESTS_PROCESSED_SUCCESSFULLY	The number of successful SOAP requests, for the system.
TOTAL_REQUESTS_RECEIVED	The number of SOAP requests, for the system.
TOTAL_UNSUCCESSFUL_REQUESTS	The number of unsuccessful SOAP requests, for the system.

Reports

Overview

The collected statistics are used for reports:

- osd ASp
- osd System

This topic covers what these reports are. For information on generating these reports see the *Report Functions* topic in *Service Management System User's Guide*.

Report Columns

The columns for the reports show:

- The reporting interval
- The peak requests for either Client ASP or system.
- The total number of requests for either Client ASP or system.

- The total number of failed requests for either Client ASP or system.

Peak requests are defined as the maximum number of requests in any on stats collection period.

For example, if the reporting interval is one hour and the stats reporting period is five minutes then the peak requests is the maximum number of requests in any one five minute period during the hour.

osd ASP report

This report provides details on SOAP requests by client ASP.

Report parameters

The table below describes the SMS parameters for this report.

Field	Description
ASP	The client ASP name, min of 1, max of 64 characters.
Reporting Period	Day, week, month reporting.
Interval	Granularity of report lines, min 10 minute intervals.
Start Date	Start date of report. Omitted reports on all statistics collected.
Hours Since	End date of report. Omitted reports on all statistics collected after Start Date.

Note: When dates are used, the format is:

YYYYMMDD[*HH24*[*MI*[*SS*]]]

Examples:

- 20090823
- 2009092312
- 200908231225
- 20090823122533

Report example

Here is an example of the OSD ASP Report.

```
         OSD ASP Report
==============================
Start Date: 01 January 2008
Finish Date: 01 January 2009
Report Type: Year
ASP: boss

17 August 2009, 21:13:23

Interval       Peak Requests Total Requests Total Failed Requests
-------------- ------------- -------------- ---------------------
20080101000000             0              0                     0
20080101002000             0              0                     0
20080101003000             0              0                     0
20080101004000             0              0                     0
20080101005000             0              0                     0
20080101010000             0              0                     0
20080101011000             0              0                     0
20080101012000             0              0                     0
20080101013000             0              0                     0
20080101015000             0              0                     0
20080101020000             0              0                     0
20080101021000             0              0                     0
```

osd System report

This report provides details on SOAP requests for the system.

Report parameters

The table below describes the SMS parameters for this report.

Field	Description
Reporting Period3	Day, week, month reporting.
Interval	Granularity of report lines, min 10 minute intervals.
Start Date	Start date of report. Omitted reports on all statistics collected.
Hours Since	End date of report. Omitted reports on all statistics collected after Start Date.

Note: When dates are used, the format is:

YYYYMMDD[*HH24*[*MI*[*SS*]]]

Examples:

- 20090823
- 2009092312
- 200908231225
- 20090823122533

Report example

```
        OSD System Report
==============================
Start Date: 01 July 2009
Finish Date: 01 August 2009
Report Type: Month

17 August 2009, 21:30:26

Interval       Peak Requests Total Requests Total Failed Requests
-------------- ------------- -------------- ---------------------
20090701000000             0              0                     0
20090701002000             0              0                     0
20090701003000             0              0                     0
20090701004000             0              0                     0
20090701005000             0              0                     0
20090701010000             0              0                     0
20090701011000             0              0                     0
20090701012000             0              0                     0
20090701013000             0              0                     0
20090701015000             0              0                     0
20090701020000             0              0                     0
```

Chapter 6

Troubleshooting

Overview

Introduction

This chapter explains the important processes on each of the server components in NCC, and describes a number of example troubleshooting methods that can help aid the troubleshooting process before you raise a support ticket.

In this chapter

This chapter contains the following topics.

Common Troubleshooting Procedures

Introduction

Refer to *System Administrator's Guide* for troubleshooting procedures common to all NCC components.

Checking current processes

You can check which processes are running using the standard UNIX command: ps. To find processes being run by Oracle software, you can grep for the string 'oper', which will display all processes being run by the application operator accounts (for example, acs_oper, ccs_oper and smf_oper).

Note: Some processes which are required for proper functioning may be run by other users, including root or the user which runs the webserver.

Example command: `ps -ef | grep oper`

For more information about the ps command, see the system documentation for the ps command.

You can also check how much of the processor a process is using by running the standard UNIX tool: top. If you have some baseline measurements, you will be able to compare it with the current load.

Example command: `top`

Tip: Some processes should only have one instance. If there are two or more instances, this may indicate a problem. For example, there will usually only be one timerIF running on each SLC.

For more information about which processes should be running on each node, check the Process List for each node in *Installation Guide*.

HTTP Error Codes

The HTTP Error Codes are listed here.

Code	HTTP Code Meaning	Meaning	Recovery
200	HTTP_CODE_OK	Normal response.	None Required.

Code	HTTP Code Meaning	Meaning	Recovery
204	HTTP_CODE_NO_CONTENT	The osdInterface received a <PING/> request. If <PING/> appears in the body of the request, the osdInterface will respond with a 204.	Remove <PING/> from the request, and retry.
400	HTTP_CODE_BAD_REQUEST	The request could not be parsed. Either the HTTP headers are invalid, the data does not fit inside internal buffers, or the body does not start with <?xml.	If the client is sending a valid SOAP request, this indicates a fault. Otherwise, correct the request syntax and retry the request.
401	HTTP_CODE_AUTHORISATION_REQUIRED	OSD requires ASPs to authenticate using Basic Access Authentication.	The client should retry the request with a valid Authorization header.
404	HTTP_CODE_NOT_FOUND	The request namespace was not found, and the request could not be processed.	Correct the namespace specification. It currently must be placed on the SOAP Operation Request element.
500	HTTP_CODE_INTERNAL_SERVER_ERROR	The server encountered an internal problem and could not continue to process the request.	Usually indicates an internal software fault. Reconciliation may be required.
503	HTTP_CODE_UNAVAILABLE	The service is unable to process the ASPs request at this time. The body of the message may offer additional information.	Correct the problem indicated in the response body and retry.

SOAP release causes

The SOAP release causes are listed here.

Cause	OSD Meaning	Meanings	Recovery
1	No such subscriber	User specified through disconnect node in control plan	
2	No such service	User specified through disconnect node in control plan	
3	Missing parameter	User specified through disconnect node in control plan	
4	Mis-typed parameter	User specified through disconnect node in control plan	
5	System error	User or software specified. The system encountered an unexpected issue processing the response. This is a catch all error, and generally indicates a software fault.	The request may or may not have been completed, reconciliation may be required once the software fault is fixed. The request may or may not work if retried.

Cause	OSD Meaning	Meanings	Recovery
6	Operation does not exist	Software specified. The operation specified in the inbound OSD request is not configured on the SLC.	Configure the operation on the SMS.
7	Operation not available	Software specified. The operation specified in the inbound OSD request is not enabled.	Enable the operation on the SMS and retry the request.
8	Invalid transaction type	Software specified. UNUSED	UNUSED
9	Transaction not found	Software specified. UNUSED	UNUSED
10	No response from ACS	Software specified – transient error only, retry permitted. At the time the response was sent, ACS had not provided a response to the interface. This will happen when the system is slow or when it is being shut down.	If it is valid to run the request multiple times, retry the request. Otherwise, use an agreed reconciliation process.
11	Too many transactions	Software specified – transient error only, retry permitted. UNUSED	UNUSED
12	Duplicate transaction	Software specified. UNUSED	UNUSED
13	Cannot Parse SOAP envelope	Software specified. The request was invalid because: **1** The XML was not valid XML **2** The service key for the operation could not be found	Ensure that the XML being presented to the service is valid, and that the SLC configuration is correct. Once this is confirmed, retry the request.
14	Operation disabled	Software specified. The ASP has insufficient permissions to run that operation.	Grant access to the operation using the SMS screens and retry the request.
63	Invalid configuration	The request was invalid because a configuration item was incorrectly configured or missing, such as a missing service entry in the **acs.conf** configuration file.	Check that OSD is correctly configured.

Note: Unless otherwise specified, these error causes are permanent failures and retry should not be attempted by the ASP.

These causes are used in the errorCode parameter of SOAP faults sent to ASPs when failures occur.

For example, if there is a profile branching node on something from Incoming Session data and the data not found branch is used, then there can be a Disconnect Node with cause 3. Then, OSD interface will send a SOAP fault with error code 3 and the ASP will know that the SOAP request has been rejected because of a missing parameter.

Chapter 7

About Installation and Removal

Overview

Introduction

This chapter provides information about the installed components for the Oracle Communications Network Charging and Control (NCC) application described in this guide. It also lists the files installed by the application that you can check for, to ensure that the application installed successfully.

In this Chapter

This chapter contains the following topics.

Installation and Removal Overview

Introduction

For information about the following requirements and tasks, see *Installation Guide*:

- NCC system requirements
- Pre-installation tasks
- Installing and removing NCC packages

OSD packages

An installation of OSD includes the following packages, on the:

- SMS:
  - osdSms
  - osdCluster (for clustered SMS
- SLC:
  - osdScp

Known issue

Once installation is complete on both nodes of a cluster, the following error may appear:

```
WARNING:  On node <host name> resource group OsdWsdlRegenerator-harg is
online but the monitor of resource OsdWsdlRegenerator-hars failed to start
```

If this happens, enter the following command line to fix the problem:

```
scswitch -z -g OsdWsdlRegenerator-harg -h The_Other_Host_Name
```

Post Install Replication

Additional replication

As part of the OSD installation the ACS_TAG_TO_PROFILE_MAPPING table replication is added to:

Operator Functions > Node Management > Table Replication > Apps > Acs_Service.

After installation, please ensure ACS_TAG_TO_PROFILE_MAPPING and ACS_PROFILE_DETAIL are replicated to all SLC nodes.

OSD Additional replication

Follow these steps to configure SMS replication to ensure all OSD definitions are available on all SLCs nodes which are running OSD.

For more information about how to complete these steps, see *Service Management System User's Guide*.

Step	Action
1	Restart SMS Java (Swing) Administration screens.
2	Open the **Table Replication** tab on the SMS Node Management screen.
3	Add the following OSD tables to the SLC nodes which have OSD installed: • OSD_CLIENT_ASP • OSD_CLIENT_ASP_ACCESS • OSD_MANDATORY_INPUT_PARAMETER • OSD_OPERATION • OSD_OPERATION_SET • OSD_PORT_LIST • OSD_PORT_LIST_ENTRY • OSD_SERVICE_HANDLE
4	Save the updated node config by clicking **Save**.
5	Click **Create Config File** update the SLCs.