Copyright

Copyright © 2013, 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, 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 on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.
Chapter 6

About Installation and Removal ........................................... 45
  Overview............................................................................. 45
  Installation and Removal Overview..................................... 45
  CDR Loader Deployment.................................................... 45
  Post-installation Configuration.......................................... 47
  Checking Removal ........................................................... 48

NCC Glossary of Terms....................................................... 51

Index ................................................................................... 59
About This Document

Scope

The scope of this document includes all the information required to install, configure and administer the USSD Gateway (UUGW phase 1) application.

Audience

This guide was written primarily for installers and System Administrators. However, sections of the document may be useful to anyone requiring an introduction to the application.

Prerequisites

A solid understanding of Unix and a familiarity with IN concepts are an essential prerequisite for safely using the information contained in this 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.

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:

- SLEE Technical Guide
- USSD GW User’s Guide
Document Conventions

Typographical Conventions

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

<table>
<thead>
<tr>
<th>Formatting convention</th>
<th>Type of information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Special Bold</td>
<td>Items you must select, such as names of tabs. Names of database tables and fields.</td>
</tr>
<tr>
<td>Italic</td>
<td>Name of a document, chapter, topic or other publication. Emphasis within text.</td>
</tr>
<tr>
<td>Button</td>
<td>The name of a button to click or a key to press. Example: To close the window, either click Close, or press Esc.</td>
</tr>
<tr>
<td>Key+Key</td>
<td>Key combinations for which the user must press and hold down one key and then press another. Example: Ctrl+P, or Alt+F4.</td>
</tr>
<tr>
<td>Monospace</td>
<td>Examples of code or standard output.</td>
</tr>
<tr>
<td>Monospace Bold</td>
<td>Text that you must enter.</td>
</tr>
<tr>
<td>Variable</td>
<td>Used to indicate variables or text that should be replaced.</td>
</tr>
<tr>
<td>Menu option &gt; menu option &gt;</td>
<td>Used to indicate the cascading menu option to be selected, or the location path of a file. Example: Operator Functions &gt; Report Functions Example: /IN/html/SMS/HelpText/</td>
</tr>
<tr>
<td>Hypertext link</td>
<td>Used to indicate a hypertext link on an HTML page.</td>
</tr>
</tbody>
</table>

Specialized terms and acronyms are defined in the Glossary at the end of this guide.
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 USSD Gateway? 1
Handset Interaction 4
Callback 6
Alarms, Statistics, Reports and EDRs 8

What is USSD Gateway?

Introduction

The USSD GW provides the following functions:

- interaction using USSD messages between the subscriber’s handset and the platform:
  - processing fast access, single string (typeahead) requests
  - presenting information to mobile users using USSD messages
  - complex interaction through navigation of menus based on user input (interactive USSD)
- IMSI Management:
  - different services can be configured for different IMSI prefixes
  - barring by IMSI or IMSI prefix
  - logging forbidden attempts to use the service, and
  - tracing for all calls from an IMSI or IMSI prefix
  - CDR Viewing screen provides full information about a call and provides EDR searching
- support for both USSD phase 1 / MAP1 and USSD phase 2 / MAP2, and
- roaming USSD Session Control:
  - separate control plans for charging and call monitoring, and
  - with Location Capabilities Pack, session can be initiated directly back to a roaming subscriber.

UIS and UPC

USSD GW is provided in two main parts:

- UIS
- UPC
Diagram

This diagram shows the components that make up the UIS part of the USSD GW service.

Components

This table describes the main components in USSD GW.

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
</table>
| ussdgw        | The ussdgw process is the main USSD GW binary. It:  
                | - provides an interface between SLEE applications (including slee_acs) and the rest of the system, and  
                | - translates between INAP and USSD.        | ussdgw (on page 33) |
| slee_acs      | The ACS process which runs control plans. | ACS Technical Guide                    |
| libupcService | libupcService is the USSD GW service library plugin for slee_acs which handles initial set up of USSD call control plans. | libupcService (on page 35) |
| libupcChassisActions | libupcChassisActions provides the functions which enable the USSD GW Feature Nodes to interact with other elements in the system. | libupcChassisActions (on page 35) |
This slee_acs plugin provides the USSD GW macro nodes.

**USSD Interactive Services Gateway**

The USSD Interactive Services Gateway (UIS) enables operators to provide interactive menu-based portal services to end users.

UIS translates between the network USSD messages received from handsets to the INAP messages used to communicate with ACS. UIS also determines the service that should handle the incoming service initiation request.

UIS enables operators to provide a range of services using USSD messages from (and to) a subscriber's handset. Interaction is configured using ACS control plans. UIS can also process fast access, single-string requests to trigger platform functionality, including:

- Subscriber account detail reports (with CCS)
- Voucher recharges (with CCS), and
- USSD Roaming callback.

**USSD Gateway Portal Service**

USSD GW's USSD Portal Service (UPC) is an optional part of USSD GW that provides extended interactivity through the UPC Portal Screens and USSD GW feature nodes.

The UPC Portal Screens are used to extend the interactive USSD menus created using the UIS screens (for example by providing menu branching).

**Handset integration**

USSD GW uses the USSD protocol as defined by GSM phase 1 & 2. This means the majority of subscribers can use the menus without needing to upgrade their handsets.

This approach is an alternative delivery mechanism to WAP, as WAP support is still limited to middle- and higher-tier handsets.
**Processing diagram**

The diagram below illustrates the possible processing stages initiated by the gateway when a message from the network (USSD message) or service interface/portal (INAP message) is received:

**Handset Interaction**

**Introduction**

There are two main methods for interacting with a handset:

1. USSD menus, and
2. Typeahead, single-string commands.
In both cases, the *ussdgw* (on page 33) process communicates back and forth with an ACS control plan. With USSD menus, the messages from the control plan are translated into USSD messages and are sent to the handset. The subscriber can then respond with another USSD message. For single string commands, ussdgw buffers the original request and responds to the messages from the control plan using each buffer in sequence.

USSD menus are created using the SMS screens. For more information about how to configure and use menus, see *USSD GW User's Guide*.

**Example call flow**

This diagram shows the call flow for a single-string handset interaction.

---

### Example call flow description

This table provides additional detail about the Example call flow diagram.

<table>
<thead>
<tr>
<th>Flow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The HLR sends a USSD message to the SLC, where it is picked up by the TCAP interface (usually SIGTRAN stack).</td>
</tr>
<tr>
<td>2</td>
<td>The TCAP interface forwards the unchanged USSD message to ussdgw across the SLEE.</td>
</tr>
</tbody>
</table>
| 3    | ussdgw parses the message, using:  
|      |   * as the initial trigger prefix and to separate each field, and  
|      |   # as a terminator.  
|      | ussdgw translates the USSD message into an INAP message, using the first field as the Service Number and forwards it to slee_acs across the SLEE.  
|      | **Note:** Service Number will not be used if a Replacement SAN is being used. For more information, see *USSD GW User's Guide*. |
| 4    | slee_acs loads the control plan for the Service Number based on standard criteria. For more information about how slee_acs determines which control plan to load, see *ACS Technical Guide*. |
Chapter 1

<table>
<thead>
<tr>
<th>Flow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>The control plan executes until it reaches an interaction node. In this example, the node is a Selection Dependant Routing node which enables the subscriber to specify which service they want to use. An INAP PACUI message is created, specifying the uugw as the srf and the specifying the announcement id 1 from the node's configuration. This message is sent to ussdgw. <strong>Warning:</strong> The srf configuration must specify uugw in the announcement and the uugw srf must also be configured in acs.conf or the message will not be received by ussdgw. For more information, see <em>srf configuration</em> (on page 12).</td>
</tr>
<tr>
<td>6</td>
<td>The ussdgw receives the PACUI and checks whether it has a buffer which contains unused data from the original USSD message. In this case it does, so it constructs an INAP CUI message using the 8 from the second field and sends it back to slee_acs.</td>
</tr>
<tr>
<td>7</td>
<td>slee_acs receives the CUI and continues the control plan as normal. In this case, the Selection Dependant Routing node routes the call to a Play Announcement node. A PA message is constructed and sent to ussdgw as described in stage 5.</td>
</tr>
<tr>
<td>8</td>
<td>ussdgw receives the PA message. In this case it has no unused buffers, so it uses the announcement id to determine what menu details to use in the USSD message it constructs and sends back to the HLR. In this case the message provides the information provided by the service selected in stage 6, and reports the date the account will expire. <strong>Note:</strong> If the interaction node specifies a number of repetitions of 127, ussdgw will not construct a message to be sent to the subscriber.</td>
</tr>
<tr>
<td>9</td>
<td>ussdgw responds to the CUI with an SRR back to slee_acs which completes the control plan.</td>
</tr>
<tr>
<td>10</td>
<td>The subscriber receives the USSD message from the HLR.</td>
</tr>
</tbody>
</table>

**Callback**

**Introduction**

USSD GW can be used to enable USSD message-initiated call back. There are a number of ways this can be configured, but the main elements are:

1. subscriber initiates the call back using a USSD message
2. the system initiates the A leg of the call, then
3. the system completes the call by initiating the B leg.

**Callback initiation**

The subscriber can initiate a callback using:

- a single string which is parsed by the ussdgw process, or
- an initial message followed by interaction defined in a control plan.

**A leg**

A-leg call initiation is done from a control plan using ACS's Call Initiation feature node. The Call Initiation node attempts to establish the A leg of the call by:

- arming the switch to inform the platform when the A party answers the call (by sending an RRBCSM (oAnswer)), and
• sending an Initiate Call Attempt (ICA) to the switch (the switch then sets up the call).

**Note:** The Call Initiation node can initiate a call with any destination number using any profile block or a hard coded value. The A leg is selected using the Call Initiation node’s configuration.

Because the A leg setup is done in a control plan, any function which is available in the control plan can be used, including:

• checking subscriber’s account state or balance, and
• normalising the calling party number.

After Call Initiation node is called, initiating control plan continues when the A leg has answered and the IDP been sent. Further processing should continue in the new call generated by the IDP.

For more information about the Call Initiation feature node, see *CPE User’s Guide*.

**B leg**

When the A party answers, the switch returns an ERBCSM (oAnswer) to the control plan and a new forked control plan starts. The new call can use any control plan functionality, including:

• monitoring the new call, and
• using a retrieved details (including MSRN) for charging.

The new forked call is responsible for connecting to the B leg (for example, by using an AT or a UATB node).

**Call back message flow**

This diagram shows a simple example of the USSD call back message flows.
Alarms, Statistics, Reports and EDRs

Alarms

USSD GW processes log alarms and notices to the syslog. They are then collected by the SMS alarms subsystem and moved to the SMS. For more information about alarms, see System Alarms.

Statistics

SMS’s statistics subsystem collects and stores the statistics on the SMS as entries in the SMF database table SMF_STATISTICS. They can then be processed further by SMS or by third party systems.

This table lists the statistics collected about USSD GW.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UIS_1</td>
<td>USSD session initiation attempt – phase 1</td>
</tr>
<tr>
<td>UIS_2</td>
<td>USSD session initiation attempt – phase 2.</td>
</tr>
<tr>
<td>UIS_3</td>
<td>Successful USSD initiation attempt (InitialDP sent to a service interface)</td>
</tr>
<tr>
<td>UIS_4</td>
<td>Message being sent to user as a result of a PACUI INAP operation from a service interface</td>
</tr>
<tr>
<td>UIS_5</td>
<td>User input as a result of an active PACUI</td>
</tr>
<tr>
<td>UIS_6</td>
<td>Fast access attempted on USSD session initiation (that is, dial ahead digits specified)</td>
</tr>
<tr>
<td>UIS_7</td>
<td>Timer Expiry (Session cut off)</td>
</tr>
<tr>
<td>UIS_8</td>
<td>Timer Expiry (SSF)</td>
</tr>
<tr>
<td>UIS_9</td>
<td>Timer Expiry (overall inactivity)</td>
</tr>
<tr>
<td>UIS_10</td>
<td>Timer Expiry (reconnect)</td>
</tr>
<tr>
<td>UIS_11</td>
<td>Timer Expiry (user inactivity)</td>
</tr>
<tr>
<td>UIS_12</td>
<td>TC-ABORT received from network</td>
</tr>
<tr>
<td>UIS_13</td>
<td>TC-ABORT received from service interface</td>
</tr>
<tr>
<td>UIS_14</td>
<td>Gateway call limiting</td>
</tr>
</tbody>
</table>

EDRs

USSD GW can log Event Data Records for some transactions. Also, an EDR is logged for each call which passes through a control plan. For more information about the EDRs logged by USSD GW see NCC Event Detail Record Reference Guide.
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.

<table>
<thead>
<tr>
<th>Component</th>
<th>Locations</th>
<th>Description</th>
<th>Further Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>ussdgw.sh</td>
<td>all SLCs</td>
<td>ussdgw.sh sets the command line parameters which configure ussdgw.</td>
<td>Gateway configuration (on page 13)</td>
</tr>
<tr>
<td>SLEE.cfg</td>
<td>all SLCs</td>
<td>SLEE.cfg sets up SLEE interfaces and applications.</td>
<td>SLEE.cfg</td>
</tr>
<tr>
<td>acs.conf</td>
<td>all SLCs</td>
<td>acs.conf configures slee_acs. This includes number normalisation.</td>
<td>Configuring acs.conf for the SLC (on page 11)</td>
</tr>
<tr>
<td>SMS screens</td>
<td>SMF database</td>
<td>The service details are configured using the SMS screens.</td>
<td>USSD GW User’s Guide</td>
</tr>
<tr>
<td>upc.conf</td>
<td>all SLCs</td>
<td>If UPC is installed, upc.conf</td>
<td>Configuring the USSD</td>
</tr>
</tbody>
</table>

Configuration Overview

Introduction

This topic covers some general information about configuring USSD GW.

For more information about configuration which must be done when USSD GW is installed, see Post-installation Configuration (on page 47).

Configuration components

USSD GW is configured by the following components:
eserv.config  all SLCs eserv.config provides date formatting for outgoing messages. (on page 19)

cdrLoader.conf cdrLoader.conf configures cdrLoader. It must be configured or cdrLoader will not start. (on page 37)

cdrIF.cfg all SLCs Configures the EDR Interface.

Multiple instances of SMSC
To configure multiple instances of the SMSC, refer to the SMSC Technical Guide.

Configuring the SLEE.cfg

Introduction
The system is configured so that USSD Gateway and associated interfaces all start together. This is performed using the /IN/service_packages/SLEE/etc/SLEE.cfg file.

Note: The directory /IN/service_packages/SLEE and all its subdirectories/files should be owned by the user acs_oper.

This can be done using: chown -R acs_oper:IN SLEE in the directory /IN/service_packages.

Editing the SLEE.cfg file
The SLEE.cfg file will be automatically edited to add the USSD Gateway components and interface entries.

Checking procedure
The SLEE.cfg configuration file is automatically updated. To check:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cd to the following directory: /IN/service_packages/SLEE/bin</td>
</tr>
<tr>
<td>2</td>
<td>An example slee.sh file: #!/bin/sh \nLD_LIBRARY_PATH=$LD_LIBRARY_PATH:/IN/service_packages/SLEE/lib \nexport LD_LIBRARY_PATH \nSHLIB_PATH=$SHLIB_PATH:/IN/service_packages/SLEE/lib \nexport SHLIB_PATH \n/IN/service_packages/SLEE/bin/sleeStartup \n/IN/service_packages/SLEE/etc/SLEE.cfg</td>
</tr>
</tbody>
</table>

Example SLEE.cfg file
Here is an example of a SLEE.cfg file that includes the USSD GW components and interface entries.

# Maximums MAXAPPLICATIONS=10 MAXSERVICES=10 MAXSERVICEHANDLES=10 MAXSERVICEKEYS=20
MAXDIALOGS=70000
MAXEVENTS=50000
MAXCALLS=25000
MAXINTERFACES=20
MAXEVENTTYPES=30
MAXCORRELATIONIDS=10000
INTERFACE=Timer timerIF /IN/service_packages/SLEE/bin UDG
INTERFACE=acsStatsLocalSLEE acsStatsLocalSLEE /IN/service_packages/ACS/bin EVENT
WATCHDOG=/IN/service_packages/SLEE/bin/ watchdog
WATCHDOG CYCLETIME=30

# Applications
APPLICATION=slee_acs slee_acs /IN/service_packages/ACS/bin 1 1
# Services
SERVICE=ACS 1 slee_acs ACS
SERVICE=ACS_Outgoing 1 slee_acs ACS_Outgoing
# Servicekeys
SERVICEKEY=INTEGER 111 ACS
SERVICEKEY=INTEGER 110 ACS_Outgoing

# USSD Gateway application and service
APPLICATION=ussdgw ussdgw.sh /IN/service_packages/UIS/bin 1 1
SERVICE=ussdgw 1 ussdgw ussdgw
SERVICEKEY=INTEGER 10 ussdgw

### Configuring acs.conf for the SLC

#### Introduction

USSD GW provides functionality which is used by the main call processing subsystem, slee_acs. slee_acs is the main binary in ACS and is configured by acs.conf.

For slee_acs to support USSD GW functionality, some configuration must be added to acs.conf.

The following pages contain a description of each section that must be changed and the acs.conf parameters that appear within that section which are relevant to USSD GW.

#### Configuring ACS to recognise hex digits

The USSD gateway can be configured to send '*' and '#' to the portal. However, the '*' and '#' is sent across the network as hex digits 'C' and 'D' respectively.

This means if ACS is used as the portal, it will need to be configured to recognise incoming hex digit 'C' as '*' and 'D' as '#'. This is achieved by adding/changing the following configuration parameters in the acsChassis section:

- DialledStarEncoding C
- DialledHashEncoding D

**Note:** The default installation of ACS has the following:

- DialledStarEncoding B
- DialledHashEncoding C

#### Checking encoding parameters

Before starting this section you must understand the layout of the ACS configuration file, acs.conf. For more details of the layout of acs.conf, refer to the *ACS Technical Guide*. 
Follow these steps to ensure that ACS recognises hex digit 'C' as '*' and 'D' as '#'.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log in to the SLC as acs_oper.</td>
</tr>
<tr>
<td>2</td>
<td>As acs_oper, edit acs.conf:</td>
</tr>
<tr>
<td></td>
<td>Example command: vi /IN/service_packages/ACS/etc/acs.conf</td>
</tr>
<tr>
<td>3</td>
<td>Set these parameters to the following:</td>
</tr>
<tr>
<td></td>
<td>DialledStarEncoding C</td>
</tr>
<tr>
<td></td>
<td>DialledHashEncoding D</td>
</tr>
</tbody>
</table>

**Notes:**
- There must be a single space before the beginning of each parameter.
- If the parameters are not found, then add them to acs.conf under the acsChassis section.

| 4 | Restart the SLEE. |

For more information about restarting the SLEE, see *SLEE Technical Guide*.

**UPC library configuration**

If the UPC part of USSD GW is being used, acs.conf must include the plugin libraries supplied by the upcScp package. A default configuration is added on installation for the following libraries:

- *libupcService* (on page 35)
- *libupcChassisActions* (on page 35), and
- *libupcMacroNodes* (on page 36).

For more information about the acs.conf entries for these libraries, see the Startup section for each binary.

**Send Buffer Node - number normalisation**

The Send Buffer node is a feature node that allows ACS to send the content of a pre-defined buffer in the form of a short message to an end-user during at runtime. It is possible to configure the origination address and destination address of the short message to normalised calling and called party numbers.

In order to use normalised calling and called party numbers either originating or destination address, normalisation needs to be configured in ACS. Whilst this is not configuration of the Send Buffer node, it is required and hence listed below.

**Note:** The calling party number is the MSISDN of the calling mobile. It is important to know the format of the MSISDN that the network passes to the USSD GW before attempting to configure ACS number normalisation.

For more information about ACS number normalisation configuration rules, see the *ACS Technical Guide*.

**srf configuration**

Control plans use Interaction nodes to send INAP messages to ussdgw as if ussdgw was a VIP or media server. In order to do this, ACS must include some specific configuration in order to work with USSD GW.

1. The interaction nodes must use announcements which have been set up to point at ussdgw instead of a normal media server. This is done by specifying announcements which use the srf of "uugw" by specifying uugw as their Resource Name in the New/Edit Announcement screen.

2. acs.conf then includes the uugw srf ids to match the announcement srf.
Example: `srf (uugw,tcapPreEnd=Y,UseETC=N,Address=,NOA=4)`

**Overview of the USSD Gateway Configuration**

**Introduction**

Exclusive configuration for the USSD Gateway is contained in the ussdgw.sh file in `/IN/service_packages/UIS/bin`.

This file is created automatically from the install script.

**Gateway configuration**

ussdgw supports these command line parameters.

Note: ussdgw is usually started by the shell script ussdgw.sh. ussdgw's configuration is usually set in the shell script. For more information about ussdgw.sh, see Startup (on page 33).

- **l** `<usr>/<pwd>`
  - Syntax: `-l <usr>/<pwd>`
  - Description: The Oracle username and password for logging into the SCP database.
  - Type: String
  - Optionality: Optional (default used if not set).
  - Allowed: 
  - Default: 
  - Notes: 
  - Example: 

- **n** `<name>`
  - Syntax: `-n <name>`
  - Description: Global Gateway Name.
  - Type: String
  - Optionality: Optional (default used if not set).
  - Allowed: Global GW config
  - Default: 
  - Notes: 
  - Example: 

- **c** `<if>`
  - Syntax: `-c <if>`
  - Description: The SLEE CDR Interface Name.
  - Type: String
  - Optionality: Optional (default used if not set).
  - Allowed: Cdr
  - Default: 
  - Notes: 
  - Example: 

-s <opt>

Syntax: 

Description: If present, sets the source for the msisdn in MAP2 messages.

Type: String

Optionality: Optional

Allowed: 

- msisdn – To use the msisdn from the body of the operation.
- msisdnref – To use the msisdnReference field.
- orig – To use the originatingReference field.
- dest – To use the destinationReference field.

Default: None

Example: 

-o <opt>

Syntax: 

Description: How to populate the MSISDN in the IDP.

Type: String

Optionality: 

Allowed: 

- imsi Use IMSI from incoming Map1BeginSubscriberActivity for the MSISDN in the IDP.
- oen Use OriginatingEntityNumber from incoming Map1BeginSubscriberActivity for the MSISDN in the IDP.
- none Do not populate the MSISDN in the IDP.

Default: 

Notes: 

Example: 

-e <opt>

Syntax: 

Description: How to populate the IMSI.

Type: String

Optionality: 

Allowed: 

- imsi Use IMSI from incoming Map1BeginSubscriberActivity for the IMSI.
- oen Use OriginatingEntityNumber from incoming Map1BeginSubscriberActivity for the IMSI.
- none Do not populate the IMSI.

Default: 

Notes: 

Example: 

-v <id>

Syntax: 

Description: The VLR announcement set id.

Type: Integer
- **m** \(<max>\)
  
  **Syntax:** \(-m \ <max>\)
  
  **Description:** The maximum number of concurrent calls allowed.
  
  **Type:** Integer
  
  **Optionality:** Optional (default used if not set).
  
  **Allowed:**
  
  **Default:** - (disabled by default)
  
  **Notes:**
  
  **Example:**

- **r** \(<opt>\)
  
  **Syntax:** \(-r \ <opt>\)
  
  **Description:** Append received PA messages to a buffer and send the contents of the buffer in the final release message.
  
  **Type:** String
  
  **Optionality:**
  
  **Allowed:** send_PA_on_Rel
  
  send_PA_on_Rel_MAP
  
  send_PA_on_Rel_MAP2
  
  **Default:**
  
  **Notes:**
  
  **Example:**

- **a**
  
  **Syntax:** \(-a\)
  
  **Description:** When present, the Release message is appended.
  
  **Type:** Boolean
  
  **Optionality:** Optional (default used if not set).
  
  **Allowed:**
  
  **Default:** Not present (unset).
  
  **Notes:**
  
  **Example:**

- **p**
  
  **Syntax:** \(-p\)
  
  **Description:** Send SpecializedResourceReport in response to PA or PACUI timeout.
  
  **Type:** Boolean
  
  **Optionality:** Optional (default used if not set).
  
  **Allowed:**
Default:  - (disabled by default)
Notes: 
Example: 

-\texttt{u}
\begin{description} \item[Syntax:] \texttt{-u} \item[Description:] Send Unicode Characters. \item[Type:] Boolean \item[Optionality:] Optional (default used if not set). \item[Allowed:] \item[Default:]  - (disabled by default) \item[Warning:] Your db character set must be UTF8 to send Unicode. \item[Example:] 

-\texttt{z <str>}
\begin{description} \item[Syntax:] \texttt{-z <str>} \item[Description:] The string to change "+"s in a number from the handset to. \item[Type:] String of zero or more digits \item[Optionality:] Optional (no replacements are made if not set). \item[Allowed:] \item[Default:] None \item[Notes:] \item[Example:] \texttt{-z 00} 
This rule would change an incoming number of +641234567890 to 00641234567890.

-w
\begin{description} \item[Syntax:] \texttt{-w} \item[Description:] Strips leading and trailing white space from the typeahead text returned in PACUI responses. \item[Type:] \item[Optionality:] Optional (no white space removal actions). \item[Allowed:] \item[Default:] None \item[Notes:] This rule would change incoming typeahead text of:
*103*1239*1239 # to 
*103*1239*1239#
\item[Example:] \texttt{-w} 

\texttt{ussdgw.sh code}
Here is the startup script code.

```
#!/bin/sh
cd /IN/service_packages/UIS/bin
exec ./ussdgw --oracle-login / --cdr-interface cdrIF
```
Chapter 2

Configuring the USSD Gateway Portal Component (UPC)

Introduction

UPC can be conceptually divided into two main components:

- Component that resides in ACS, which is started and controlled by ACS.
- Component that is controlled by the SLEE directly.

The part of UPC that resides in ACS is configured via a single configuration file, upc.conf. This file resides in /IN/service_packages/UPC/etc and is owned by upc_oper.

Default upc.conf file

Here is the default upc.conf file that is delivered in the upcScp package.

```plaintext
# Service loader configuration
cpcServiceLoader

    # This is the cause that will be used in an INAP ReleaseCall operation
    # when no call plan could be found for the incoming call
    noCallPlanReleaseCause 31

    # This is the default language that will be used when no user-specific
    # language could be determined
    defaultLanguageId 1

# DO NOT DELETE the ': ' below!
:

cpcServiceLoader parameters

As this is the ACS UPC component configuration, the acs.conf-style configuration is used. This means all configuration must belong to a section, and a configuration section ends with a single ':'. The '#' in the beginning of a line indicates a comment, and is ignored by the configuration parser. Actual configuration are done via key/value pair with a space in between followed a new line.

Currently the only sub-component in the ACS UPC module that requires configuration is the UPC service loader, and the section is named "cpcServiceLoader".

There are a number of configuration parameters for the UPC service loader, listed below:

noCallPlanReleaseCause

**Syntax:**

noCallPlanReleaseCause <value>

**Description:**

The UPC service loader attempts to load a control plan based on the dialled number/replacement SAN that is sent by the gateway in the beginning of a call. The UPC service loader will release the call with the configured release cause if the dialled number/SAN to control plan mapping is not defined.

**Type:**

Integer

**Optionality:**

Mandatory

**Allowed:**

Valid Release Cause value. Refer to the ACS Technical Guide.

**Default:**

Notes:

**Example:**

noCallPlanReleaseCause 31
defaultLanguageId
Syntax: \texttt{defaultLanguageId} \texttt{<value>}
Description: The language ID is used in conjunction with announcement IDs to achieve transparent multi-lingual announcement playing. This configures the default language ID when an end-user dials a USSD call and no user specific language could be determined.
Type: Integer
Optionality: Mandatory
Allowed: 
Default: 
Note: The language ID configured needs to match the ACS and UIS language ID configuration.
Example: defaultLanguageId 1

noMsisdnReleaseCause
Syntax: \texttt{noMsisdnReleaseCause} \texttt{<value>}
Description: If a call is received without a MSISDN (mapped in the Calling Party Number in the IDP), the UPC service loader will release the call with the configured release cause.
Type: Integer
Optionality: 
Allowed: Valid release cause value. Refer to the ACS Technical Guide.
Default: 
Notes: 
Example: 

smscInterfaceName
Syntax: \texttt{smscInterfaceName} \texttt{<name>}
Description: Used by the Send Buffer node if the SMSC SLEE Handle is not provisioned in the Send Buffer node.
Type: String
Optionality: Optional
Allowed: 
Default: smscIF
If the SMSC IF name is not provisioned in the Send Buffer screen, and the name is not provisioned in the Service Loader configuration the default name will be set by the Send Buffer node.
Notes: 
Example: 

smscFromAddress
Syntax: \texttt{smscFromAddress} \texttt{<value>}
Description: Used by the Send Buffer node to populate the Originating Address field of the UPC message if one is not provisioned in the Send Buffer screen.
Type: String
Optionality: Optional
Allowed: 

Release calls with no MSISDN instructions

Follow these steps to modify upc.conf to disconnect calls with no MSISDN (such as MAP 1 calls). This is optional.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log in as upc_oper # su - upc_oper</td>
</tr>
</tbody>
</table>
| 2    | Edit upc.conf:  
  ```bash
  $ vi /IN/service_packages/UPC/etc/upc.conf
  ```  |
| 3    | Add noMsisdnReleaseCause `<value>` within the upcServiceLoader (on page 17) section. Note that indentation may be added for human-readability. |
| 4    | Restart the SLEE. |
| 5    | If a call release cause has been configured in the UPC.conf (that is -noCallPlanReleaseCause or noMsisdnReleaseCause), the following will also need to be configured, enabling the correct text message to be sent back to the user.  
  The Status Info tab allows you to map status values to move meaningful status messages.  
  Using the SMS screen select Service>USSD Gateway>Menu & Status>Config and select the Status Info tab. Add a new name and add a value that is equal to the number used for the release cause in the upc.conf file.  
  For further details refer to the USSD Gateway User Guide. |
| 6    | The Status Language tab allows you to set language specific status text for a given status.  
  Select Service>USSD Gateway>Menu & Status>Display and select the Status Language tab.  
  Select the menu just created and add the text required to be displayed to the user. |

eserv.config Configuration

Introduction

The eserv.config file is a shared configuration file, from which many 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 up 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 may be used, as in this example:

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

or

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

or

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

Location of eserv.config

By default, UPC will read its configuration from the LCA section of:

```
/IN/service_packages/eserv.config
```

To override the default location, use the ESERV_CONFIG_FILE environmental variable.

```
ESERV_CONFIG_FILE
```

**Syntax:**

```
ESERV_CONFIG_FILE = "path/file"
```

**Description:**
The directory eserv.config configuration file will be read from.

**Type:**

String

**Optionality:**
Optional (default used if not set).

**Allowed:**

**Default:**

```
/IN/service_packages/eserv.config
```

**Notes:**

**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 will cause 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 will ensure you have a working copy to which you can return.
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.

Warning: This file is not intended to be changed by the user. Please contact Oracle support with your queries.

Response Date and Time

Response date and time format

Responses to the USSD queries are based on the chosen language of the subscriber making the query. The USSD responses may contain date and time information. This section of the eserv.config file allows the format of the date and time to be configured based on the chosen language of the subscriber.

Parameters

Here are the parameters supported by ussdgw (on page 33) in the UIS.DateAndTime section of the configuration file.

DaysOfWeek

Syntax: DaysOfWeek = {
    <lang> = {
        Full = [ <config> ]
        Abbv = [ <config> ]
    }
    [...]  
}

Description: The names of the days of weeks in various configured languages used in outgoing USSD messages.

Type: Array

Optionality: Optional (defaults used if not set).

Allowed: As set by Full (on page 22) and Abbv (on page 22) parameters.

Default: As set by Full (on page 22) and Abbv (on page 22) parameters.

Notes: There must an entry for each of all the 7 days of the week or an alarm will be logged when ussdgw (on page 33) starts up and the default OS language will be used instead of the chosen subscriber language.

For more information about how these values are used, see USSD GW User Guide, Languages tab.

Example: DaysOfWeek = {
    English = {
        Full = [ "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday" ]
        Abbv = [ "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" ]
    }
    Bahasa = {
    }
}
Chapter 2

Language
Syntax: For an example of how to use this parameter, see DaysOfWeek (on page 21).
Description: Name of the language.
Type: String
Optionality: Optional (default used if not set).
Allowed: Must match a language defined on the Language tab of the USSD Gateway Base Configuration Screen.
Default:
Notes: For more information about the USSD Gateway Base Configuration Screen, see USSD GW User Guide.
Example: For an example of this parameter used in context, see Examples (on page 24).

Full
Syntax: Full = [ "<sun>", "<mon>", "<tue>", "<wed>", "<thu>", "<fri>", "<sat>" ]
Description: Full names of the days of the week in the specified language beginning with “Sunday” and ending with “Saturday”.
Type: Array of Strings
Optionality: Optional (defaults used if not set).
Allowed: Default: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday
Notes: Used to define the %A variable in the Data Format field on the Language tab. There must an entry for each of all the 7 days of the week.
Example: For an example of this parameter used in context, see Examples (on page 24).

Abbv
Syntax: Abbv = [ "<sun>", "<mon>", "<tue>", "<wed>", "<thu>", "<fri>", "<sat>" ]
Description: Abbreviated names of the days of the week in the specified language, beginning with “Sun” and ending with “Sat”.
Type: Array of Strings
Optionality: Optional (defaults used if not set).
Allowed: Default: Sun, Mon, Tue, Wed, Thu, Fri, Sat
Notes: Used to define the %a variable in the Data Format field on the Language tab. There must an entry for each of all the 7 days of the week.
Example: For an example of this parameter used in context, see Examples (on page 24).

Months
Syntax: Months = {
    <lang> = {
        Full = [ <config> ]
        Abvv = [ <config> ]
    }
    [...]}
Description: The names of the months of the year in various configured languages used in outgoing USSD messages.
Chapter 2

Type: Array
Optionality: Optional (defaults used if not set).
Allowed: As set by Full (on page 23) and Abbv (on page 24) parameters.
Default: There must an entry for each of all the 12 months of the year or an alarm will be logged when ussdgw (on page 33) starts up and the default language will be used.
Notes: For more information about how these values are used, see USSD GW User Guide, Languages tab.

Example: Months = {
  English = {
  }
  Bahasa = {
  }
}

Language
Syntax: For an example of how to use this parameter, see Months (on page 22).
Description: Name of the language.
Type: String
Optionality: Optional (default used if not set).
Allowed: Must match a language defined on the Language tab of the USSD Gateway Base Configuration Screen.
Default:
Notes: For more information about the USSD Gateway Base Configuration Screen, see USSD GW User Guide.
Example: For an example of this parameter used in context, see Examples (on page 24).

Full
Syntax: Full = [ "<jan>", "<feb>", "<mar>", "<apr>", "<may>", "<jun>", "<jul>", "<aug>", "<sep>", "<oct>", "<nov>", "<dec>" ]
Description: Full names of the months of the year in the specified language beginning with “January” and ending with “December”.
Type: Array of Strings
Optionality: Optional (defaults used if not set).
Allowed: January, February, March, April, May, June, July, August, September, October, November, December
Default:
Notes: Used to define the %B variable in the Data Format field on the Language tab.
There must an entry for each of all the 12 months of the year.
Example: For an example of this parameter used in context, see Examples (on page 24).

Abbreviation
Syntax: `Full = [ "<jan>", "<feb>", "<mar>", "<apr>", "<may>", "<jun>", "<jul>", "<aug>", "<sep>", "<oct>", "<nov>", "<dec>" ]`
Description: Abbreviated names of the months of the year in the specified language beginning with "Jan" and ending with "Dec".
Type: Array of Strings
Optionality: Optional (defaults used if not set).
Allowed: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
Notes: Used to define the %b variable in the Data Format field on the Language tab. There must an entry for each of all the 12 months of the year.
Example: For an example of this parameter used in context, see Examples (on page 24).

Examples

Example 1
This text shows an example of 1 Language:

- **Primary Language** = Bahasa
- **Other Languages** = None

```
DateAndTime = {
  DaysOfWeek = {
    Bahasa = {
    }
  }
  Months = {
    Bahasa = {
    }
  }
}
```

Example 2
This text shows an example of 4 Languages:

- **Primary Language** = Japanese
- **Other Languages** = English, Bahasa, Polish

```
DateAndTime = {
  DaysOfWeek = {
    English = {
      Full = [ "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday" ]
      Abbv = [ "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" ]
    }
    Bahasa = {
    }
  }
}
```
EDR Section

Introduction

The ussdgw application no longer relies on cdrIF to generate EDRs. EDRs are now generated directly by the ussdgw application. To enable EDR generation, the parameters listed in this section must be configured.

Example config

The following parameters show the EDR section of the eserv.config file:

```json
UIS = {
    EDR = {
        flushPeriod = 1800    # 30 minutes
        filePrefix = "UIS"
        tagPrefix = "UIS"
    }
}
```
Parameters

These are the EDR parameters:

**destDir**

Syntax: 
```java
destDir = "<directory>"
```

Description: Base file store directory for completed EDR files.

Type: String

Optionality: Optional (default used if not set).

Default:
```java
"/IN/service_packages/UIS/edr/closed"
```

Notes: **Warning:** The directory specified here must exist and have the correct permissions for the user that executes the ussdgw process.

Example: 
```java
destDir = "/IN/service_packages/UIS/edr/closed"
```

**filePrefix**

Syntax: 
```java
filePrefix = "<>"
```

Description: Base filename used to create EDR log files.

Type: String

Optionality: Optional (default used if not set).

Default: 
```java
"" - null value
```

Notes: Could be used to easily indicate which process the EDR file was generated by.

Example: 
```java
filePrefix = "UIS" would give a file name of:
"UIS" + pid + "YYYYMMDDHHMMSS.cdr"
```

**flushPeriod**

Syntax: 
```java
flushPeriod = <seconds>
```

Description: How long (in seconds) before closing the current EDR file and moving to the Destination EDR Directory (ref. destDir).

Type: Integer

Optionality: Optional (default used if not set).

Allowed: Any positive integer.

Default: 
```java
0 - A value of zero indicates that no EDR files will be generated.
```
Chapter 2

Notes: A recommended value for flushPeriod would be 600 (10 minutes) or larger. Setting this value too small (e.g. less than 2 or 3 minutes) may not be optimum with respect to system performance as this would cause EDR files to be generated too often.

Used in conjunction with the maxNum parameter, a value of say 1800 would allow EDR files to be generated every 30 minutes or earlier if the number of EDRs in the current file exceeds the maxNum value.

**Warning:** If flushPeriod is not set (or set to 0), no EDR files will be generated even if the Cdr Flag check boxes on the SMS Screens are ticked.

Example: flushPeriod = 1800

**maxNum**

Syntax: maxNum = <value>

Description: Max number of EDRs per file.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: Any positive integer.

Default: 10000

Notes: Used in conjunction with the flushPeriod parameter.

Example: maxNum = 6000

**tagPrefix**

Syntax: tagPrefix = "<value>"

Description: A string that will be inserted at the start of each row in the EDR files generated by USSD Gateway.

Type: String

Optionality: Optional (default used if not set).

Allowed:

Default: "" - null value

Notes: 

Example: tagPrefix = "UIS"

The EDR file contents would look like:

UIS|callID=1160640033|type=0|IMSI=555551234567891|...etc...
UIS|callID=1160640034|type=0|IMSI=555551234567891|...etc...
UIS|callID=1160640035|type=0|IMSI=555551234567891|...etc...
UIS|callID=1160640036|type=0|IMSI=555551234567891|...etc...

**tempDir**

Syntax: tempDir = "<directory>"

Description: Temporary directory for working EDR files.

Type: String

Optionality: Optional (default used if not set).

Allowed:

Default: "/IN/service_packages/UIS/edr/current"

Notes: **Warning:** The directory specified here must exist and have the correct permissions for the user that executes the ussdgw process.
Example: \texttt{tempDir = "/IN/service\_packages/UIS/edr/current"}

timestampFormat

Syntax: \texttt{timestampFormat = "<symbolic parameters>"}

Description: The format of timestamps shown in the EDR files.

Type: String

Optionality: Optional (default used if not set).

Allowed: Any valid formatting parameters as per Unix command \texttt{man strftime}.

Default: 

\texttt{"%Y-%m-%d \%T"}

Notes: The format is as described in the Unix command \texttt{man strftime}, with the additions for specifying microseconds \texttt{[usec:x]}.

Where \( x \) is an integer between 1 and 6 (inclusive) which specifies the number of microsecond digits required.

Example: \texttt{timestampFormat = "%Y-%m-%d \%T[usec:6]"}

With a timestamp of 2010-02-18 03:59:09 59975 microseconds the timestamp will be output in the EDR as:

2010-02-18 03:59:09.059975

With \texttt{[usec 4]}, same timestamp will appear as:

2010-02-18 03:59:09.5997

Introduction

The EDR Interface (cdrIF) handles and stores all of the customer records for all requests made to the USSD GW service. The interface must be configured to process files with a specific structure, and store them in a specific location.

For more information about cdrIF, see \textit{SLEE CDR Interface Technical Guide}.

Example cdrIF.cfg configuration file

Here is an example cdrIF.cfg configuration file.

Defaults {
    TempDirectory="/IN/cdr/temp"
    FileSize=4096
    FileDirectory="/IN/cdr/default"
}

RecordDef "CDR1" {
    FileDesc {
        FileDirectory="/IN/cdr/complete"
        FileName="%O(%y%M%d%H%M%S)-%C(%y%M%d%H%M%S).cdr1"
        FileHeader="%O"
        FileFooter="%C"
        RowHeader="Entry "
        FileSize=1024
        RowTrailer="\n"
        ColumnSeparator="",
        RemoveNullColumns=true
    }

    ColumnDef {
        AccountName    "%s"    ""
        CreditBefore   "%.2f"   "0"
        CallType       "%d"    "0"
        TimeStarted    "%H%M%S-%Y%M%d"   "0"
        TimeEnded      "%H%M%S-%Y%M%d"   "0"
    }
}
RecordDef "UIS" {
  FileDesc {
    FileDirectory="/IN/cdr/usdd/"
    FileName="%O(%y%M%d%h%m%s)-%C(%y%M%d%h%m%s).cdr"
    RowHeader="UIS"
    RowTrailer="\n"
    ColumnSeperator="|"
    RemoveNullColumns=false
  }
  ColumnDef {
    callID           "callID=%d"            "callID="
    type             "type=%d"             "type="
    IMSI             "IMSI=%s"             "IMSI="
    MSISDN           "MSISDN=%s"           "MSISDN="
    dialledString    "originalDialledString=%s" "originalDialledString="
    SAN              "SAN=%s"              "SAN="
    serviceIf        "serviceIf=%d"        "serviceIf="
    operator         "operator=%d"         "operator="
    mapPhase         "mapPhase=%d"          "mapPhase="
    fastAccess       "fastAccess=%s"       "fastAccess=N"
    trace            "trace=%s"            "trace="
    interactions     "interactions=%d"      "interactions=0"
    alarms           "alarms=%s"            "alarms="
    sessionTimeout   "sessionTimeout=%s"    "sessionTimeout=0"
    inactivityTimeout "inactivityTimeout=%s" "inactivityTimeout=0"
    reconnectTimeout "reconnectTimeout=%s"  "reconnectTimeout=0"
    prepostpay       "prepostpay=%s"       "prepostpay="
    vlr              "vlr=%s"              "vlr="
    startTime        "startTime=%s"         "startTime="
    sessionDuration  "sessionDuration=%d"   "sessionDuration="
  }
}

Configuring the XML Interface and Enabling Tracing

xmlIF.cfg configuration

The XML interface configuration file is used by the XML interface to determine the port, ip address and response time for the XML server. During the installation of the UPC package the installation script will prompt the user for XML server parameters. These parameters will be saved in the xmlIF.cfg file described below. This file is also used to set the XML tracing parameters, which are used to enable/disable and direct the XML tracing file.

ip

Syntax:       ip <value>
Description:  IP address of content provider.
Type:         String
Optionality:  Mandatory
Allowed:      IP address in standard format
Default:      
Notes:        
Example:      ip 192.1.2.64
port
Syntax: port <value>
Description: Port on the machine in which the requests and responses are read and written to.
Type: Integer
Optionality: Mandatory
Allowed: 
Default: 9999
Notes: 
Example: port 9999

timeout
Syntax: timeout <value>
Description: The response timeout from the content provider, in ms.
Type: Integer
Optionality: Mandatory
Allowed: 
Default: 3000
Notes: 
Example: timeout 3000

xmlfile
Syntax: xmlfile <path>
Description: This defines the file which contains the tag pairs, needed in constructing the request sent to the XML Service provider.
Type: String
Optionality: Mandatory
Allowed: Valid path
Default: 
Notes: 
Example: xmlfile /IN/service_packages/UPC/etc/<file 1>

tracingPath
Syntax: tracingPath <path>
Description: Defines the directory in which the trace file will be written to.
Type: String
Optionality: Optional
Allowed: valid path
Default: /IN/service_packages/UPC/tmp
Notes: 
Example: tracingPath /IN/service_packages/UPC/tmp

tracingEnabled
Syntax: tracingEnabled <true |false>
Description: Defines if tracing is enabled or disabled.
Type: boolean
Optionality: Optional
Allowed: true, false
Default: false
Notes:
Example: tracingEnabled true

keepalive
Syntax: keepalive <true |false>
Description: Keeps the connection with the XML server alive.
Type: boolean
Optionality: Optional
Allowed: true, false
Default: true
Notes:
Example: keepalive true

Example xmlIF.cfg
Here is an example xmlIF.cfg file.

```
ip 192.1.2.64
port 9999
timeout 3000
xmlfile /IN/service_packages/UPC/etc/<file 1>
xmlfile /IN/service_packages/UPC/etc/<file 2>
tracingPath /IN/service_packages/UPC/tmp
tracingEnabled true
keepalive true
END
```

XML script configuration
The XML scripts are individually written. They are placed in the IN/service_packages/UPC/etc directory.

XML interface tracing
The XML Interface can create a Trace log file, which is used to monitor debug messages from the XML Interface. It is possible to switch on the output of trace events by sending the XML Interface process a signal at run time or by a specification within the XML Interface configuration file at start up.

These trace events will be written to a pre-defined trace log file. It is also possible to specify the location of this file within the XML Interface configuration file. The following describes the configuration and viewing of the trace logs generated by the XML Interface.

Configuration
At start up the XML interface reads the tracing configuration, if it has been set in the xmlIF.cfg file. Otherwise the default values are set to tracing switched off and the trace file is created in /IN/service_packages/UPC/tmp. If this directory does not exist, it will dump the file in /tmp.
Switching on trace at start up:

<table>
<thead>
<tr>
<th>tracingPath</th>
<th>The directory where the trace can be created.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tracingEnabled</td>
<td>this will switch on or off the tracing at start up, or when xmlIF.cfg is reread, this can be set to true or false.</td>
</tr>
</tbody>
</table>

For example, add the following two lines to /IN/service_packages/UPC/xmlIF.cfg:

```plaintext
tracingPath /IN/service_packages/UPC/tmp
tracingEnabled true
```

Switching trace on or off, after start up

Follow these steps to switch trace on and off, after start up.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The control of the trace can be achieved by sending the xmlInterface process a known signal. The following describes the functionality of these signals.</td>
</tr>
<tr>
<td></td>
<td>• HUP - This will toggle the trace to either on or off.</td>
</tr>
<tr>
<td></td>
<td>• USR1 - This will cause the xmlInterface to reread the xmlIF.cfg file</td>
</tr>
<tr>
<td></td>
<td>• USR2 - This will cause the xmlInterface to disconnect and reconnect to the XML Server.</td>
</tr>
<tr>
<td>2</td>
<td>To use these signals with the XML interface, first identify the pid for the XML interface.</td>
</tr>
<tr>
<td>3</td>
<td>Send the specified signal to the XML interface process using the kill command. (For full details refer to the man kill pages section 1).</td>
</tr>
</tbody>
</table>

**Example:**

```bash
$ kill -USR2 <pid>
```

The trace file

The trace file will be located in a directory and a new Trace file will be created each day at midnight. The file will take the form:

- xmlTrace_<date>.log. For example:
  - xmlTrace_20030622.log
Overview

Introduction

This chapter explains the processes which run automatically as part of the application. These processes are started automatically by one of the following:

- inittab
- crontab
- 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.

ussdgw 33
UssdMfileD 34
libupcService 35
libupcChassisActions 35
libupcMacroNodes 36
cdrLoader 36

ussdgw

Purpose

The ussdgw process is the main USSD GW binary. It:

- translates incoming USSD messages into INAP messages which are passed to a SLEE application (such as slee_acs)
- determines which service key an incoming USSD message should trigger to in the SLEE, and
- translates INAP play announcements and PACUI messages into USSD messages and forwards them to the external interface.

Location

This binary is located on SLCs.

Startup

This task is started by the SLEE, by lines like the following in SLEE.cfg:

APPLICATION=ussdgw ussdgw.sh /IN/service_packages/UIS/bin 1 1
SERVICE=ussdgw 1 ussdgw ussdgw
SERVICEKEY=INTEGER 10 ussdgw
Notes:
- Actual value and startup script name may vary.
- For more information about this SLEE.cfg configuration, see *SLEE Technical Guide*.

**Configuration**

ussdgw is configured using the command line. For more information about the available parameters, see *Gateway configuration* (on page 13) and *eserv.config Configuration* (on page 19).

**UssdMfileD**

**Purpose**

UssdMfileD maintains all USSD Gateway MFiles. It is installed with UIS.

*Note:* The MFiles contain a sub-set of the configuration data (such as triggering rules) entered through the UPC and UIS system management screens. This data is stored in a form optimised for fast lookup by *ussdgw* (on page 33).

**Location**

This binary is located on SLCs.

**Startup**

This task is started twice (by entries uis0 and uis1 in the inittab). Each entry uses a different startup shell script. They are:

/IN/service_packages/UIS/bin/uisMfileOPStartup.sh
/IN/service_packages/UIS/bin/uisMfileTRStartup.sh

**Configuration**

UssdMfileD supports these parameters from command line:

```bash
UssdMfileD -user <uid>/<pwd> -name <name>
```

**-user**

**Syntax:** `-user <usr>/<pwd>`

**Description:** The oracle userid and password to log into the database.

**Type:** String

**Optionality:** Optional (default used if not set).

**Allowed:**

**Default:** `/`

**Notes:**

**Example:**

```
-username
```

**-name**

**Syntax:** `-name <name>`

**Description:** The filename of the MFile.

**Type:**

**Optionality:** Mandatory

**Allowed:**

-
Default: None

Notes:

Example:

UssdMfileD -user smf/smf -name UIS_OPERATOR_INFO_MFILE
UssdMfileD -user smf/smf -name UIS_SVC_TRIGGER_MFILE

Output

UssdMfileD writes alarms and other messages to the syslog and to:

/IN/service_packages/UIS/tmp/uisMfileOP.log
/IN/service_packages/UIS/tmp/uisMfileTR.log

libupcService

Purpose

libupcService is the USSD GW service library plugin for slee_acs which handles initial set up of USSD call control plans. It:

- sets up USSD GW call processing (including populating the call context from the IDP), and
- used the eserv.config and USSD GW screens configuration to determine the correct control plan to load and run from cache.

Location

This library is located on SLCs.

Startup

If libupcService is configured in acs.conf, it is made available to slee_acs when slee_acs is initialised. It is included in the acsChassis section of acs.conf in a ServiceEntry.

acsChassis
ServiceEntry (UPC,C,c,libupcService.so)

Configuration - libupcService - 1.0- ; #lgcy

libupcService is configured in the upcServiceLoader section of the upc.conf file.

For more information about this configuration, see Configuring the USSD Gateway Portal Component (UPC) (on page 17).

libupcChassisActions

Purpose

libupcChassisActions provides the functions which enable the USSD GW Feature Nodes to interact with other elements in the system, including ussdgw.

Location

This library is located on SLCs.
### Startup

If `libupcChassisActions` is configured in `acs.conf`, it is made available to `slee_acs` when `slee_acs` is initialised. It is included in the `acsChassis` section of `acs.conf` in a `ChassisPlugin` entry.

```
acsChassis
  ChassisPlugin libupcChassisActions.so
```

### Configuration

This binary has no specific configuration.

### `libupcMacroNodes`

#### Purpose

This `slee_acs` plugin provides the USSD GW macro nodes. There are no configuration file settings for these macro nodes, they are all configured in the Control Plan Editor node configuration screens.

For more information about the feature nodes provided by this library, see *USSD GW User's Guide*.

For more information about macro node libraries, see *ACS Technical Guide*.

For more information about the CPE, see *CPE User's Guide*.

#### Location

This library is located on SLCs.

#### Startup

If `libupcMacroNodes` is configured in `acs.conf`, it is made available to `slee_acs` when `slee_acs` is initialised. It is included in the `acsChassis` section of `acs.conf` in a `MacroNodePluginFile` entry as follows:

```
acsChassis
  MacroNodePluginFile libupcMacroNodes.so
```

### Configuration

This binary has no specific configuration.

### `cdrLoader`

#### Purpose

cdrLoader reads EDR files or standard input and inserts records into SMF database.

Required to view EDRs in CDR Viewer screen.

#### Location

This binary is located on SMSs.

#### Startup

This task is run in the crontab for `acs_oper`, by default every minute. It is scheduled as the following script:

```
The script runs the cdrLoader process with set parameters. cdrLoaderCron.sh will not start another cdrLoader process if one is already running.

**Configuration**

cdrLoader supports these parameters from cdrLoader.conf.

```plaintext
username=<usr>
password=<pwd>
sname=<nsname>
```

*Note: cdrLoader will not start if cdrLoader.conf cannot be found.*

**username**

- **Syntax:** `username=<usr>`
- **Description:** The userid to use for logging into the SMF database.
- **Type:** Optional
- **Allowed:**
- **Default:**
- **Notes:**
- **Example:** `username=SMF`

**password**

- **Syntax:** `password=<pwd>`
- **Description:** The password to use for logging into the SMF database.
- **Type:**
- **Optionality:**
- **Allowed:**
- **Default:**
- **Notes:**
- **Example:** `password=PWD`

**nsname**

- **Syntax:** `nsname=<nsname>`
- **Description:**
- **Type:** Optional.
- **Allowed:**
- **Default:**
- **Notes:** Operator can also set nsname as it appears in tnsnames.ora
- **Example:**
Overview

Introduction
This chapter explains the procedures for administering the USSD Gateway application.

In this chapter
This chapter contains the following topics.
Starting and Stopping the USSD Gateway

Starting and Stopping the USSD Gateway

Introduction
This topic explains how to start or stop the USSD Gateway application.

Starting the SLEE
Follow these steps to start the automated shell script.

Note: You must be logged in as the user acs_oper.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Type /IN/service_packages/SLEE/bin/slee.sh</td>
</tr>
</tbody>
</table>

Result: This shell script starts the slee_acs and the associated interfaces ussdgw, timer IF and cdrIF.

The stdout and stderr from slee.sh will appear on the screen, so if this screen is closed the output will no longer be viewable. If this information is required then redirect output to a file, e.g. slee.sh > sleeout.log

Startup output
When the SLEE service starts various information is presented on stdout and the syslog.

Stopping the USSD Gateway service
Follow these steps to stop the USSD Gateway service.

Note: You must be logged in as the user acs_oper.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Type /IN/service_packages/SLEE/bin/stop.sh</td>
</tr>
</tbody>
</table>

Note: It also recommended to run a ./clean following the stop.
If the SLEE_FILE variable is being used it must be visible to the stop program. If it is not visible, the program will not be able to clear the shared memory and will exit with error 3005.

Note: If the service has stopped for any abnormal reasons, a manual cleanup should be performed, i.e. ps -fu acs_oper to find the remaining processes, then kill <pid> each one. The shared memory should be checked using ipcs | grep abs, then remove acs_oper owned ones using ipcrm.
Overview

Introduction

This chapter explains the important processes on each of the server components in the NCC, and a number of example troubleshooting methods which will help aid the troubleshooting process before raising a support ticket.

In this chapter

This chapter contains the following topics.

- Common Troubleshooting Procedures 41
- Scenarios 41

Common Troubleshooting Procedures

Introduction

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

Debug

Logging (debugging) can be enabled on an IMSI basis. The output from specific debugs are written to files with names derived from what is being debugged.

Scenarios

Checking the service

Refer to the table below for a list of possible problems and the course of action required to fix.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>The service does not appear to be running as expected</td>
<td>Check that the service is actually running. If it is, using the Unix <code>ps</code> command, you should get a response similar to the following: <code># ps -fu acs_oper acs_oper 1975 1 0 11:38:07 pts/7 0:00 ./ussdgw - -oracle-login / --cdr-interface cdrIF</code> The main thing to note here is that the <code>ussdgw</code> (on page 33) process is running. If it is not running, then the SLEE has not been started.</td>
</tr>
<tr>
<td>EDRs are not being written</td>
<td>Check that the CDR flag in the Gateway configuration has been set, as described in the <em>USSD GW User's Guide</em>. Check that the user who runs the SLEE (acs_oper) has permission to write to the <code>/IN/cdr/temp and /IN/cdr/ussd directories</code>. Once the permissions are changed, the cdrIF will start up automatically and EDRs will be written to a file in the <code>/IN/cdr/temp</code> directory.</td>
</tr>
<tr>
<td>Problem</td>
<td>Remedy</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>Alarms are not being logged</td>
<td>To check for alarms, use: <code>tail -f /var/adm/syslog/syslog.log</code>. If alarms are not being logged, then the SMS alarm subsystem has not been properly installed. Contact your system administrator.</td>
</tr>
<tr>
<td>When starting the SLEE, the following error appears: SLEE Exception (1005) in sleeUnixSemaphore.cc at 114 by process id 2537</td>
<td>You need to increase the number of semaphores and rebuild the kernel. SEMMNS SEMMNI SEMMSL Then rebuild the kernel using the <code>sam</code> command, and reboot the machine.</td>
</tr>
</tbody>
</table>
| When running the SLEE, the following error appears when passing calls: May 1 14:28:09 cmnError(20187) NOTICE: smsRecordStats: Statistic not found 'UIS.UIS_5' | Check: in sqlplus: 
```
SQL> select count(*) from smf_statistics_defn;
COUNT(*)  
----------
49
```
and in smsStatsDaemon.log:
```
# cd /IN/service_packages/SMS/tmp
# view smsStatsDaemon.log
It will include a line similar to this one:
smsStatsDaemon: Adding up to 49 entries (some possibly SYSTEM)
• That the number of entries is the same. If they are not, kill the smsStatsDaemon process, restarting it. Then check again and the two numbers should match. |
| When starting the SLEE, the following error appears: No Output because output file is Null | You need to verify that the user who started the SLEE (acs_oper) has permission to write to the directories required for EDRs, statistics. CDR subsystem: /IN/cdr/ussd Stats subsystem: /IN/service_packages/SMS/stats |
| Running a call to the USSD Gateway product and it fails with “Service Trigger Undetermined” | One of the `UssdMfileD` (on page 34) processes may not be running. 
```
ps -ef | grep Ussd
uis_oper 9198 1 0 Mar 11 ? 0:00 /IN/service_packages/UIS/bin/UssdMfileD -name UIS_SVC_TRIGGER_MFILE -user /uis_oper 16284 1 0 15:38:47 ? 0:00 /IN/service_packages/UIS/bin/UssdMfileD -name UIS_OPERATOR_INFO_MFILE -user /
```
If either of these daemons is not running they need to be started. If they are running they may need to be restarted by killing the existing processes. If they are running and current, then check configuration of service triggers in USSD GW screens. |
<p>| When removing the upcSms package the following is displayed: | Abort the uninstallation. Run the SMS screens ACS CPE export the control plans containing these nodes. Go to the Resources screen in ACS and un-associate any service numbers with these call plans. |</p>
<table>
<thead>
<tr>
<th>Problem</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>./upcSms.unconf.sh: ./checkCanUnconfigure.sh: not found</td>
<td>Return to the CPE delete the control plan data for these call plans and then delete the structure for these control plans. You can now return to the uninstallation of the package.</td>
</tr>
<tr>
<td>* The following macro nodes are still in use by one or more call plans ...</td>
<td></td>
</tr>
<tr>
<td>* Version Branching</td>
<td></td>
</tr>
<tr>
<td>* User Selection</td>
<td></td>
</tr>
<tr>
<td>* Language Setting</td>
<td></td>
</tr>
<tr>
<td>* User Input</td>
<td></td>
</tr>
<tr>
<td>* Send Buffer</td>
<td></td>
</tr>
<tr>
<td>* Call plans using these nodes should be exported first. Abort uninstallation ? [y,n,?]</td>
<td></td>
</tr>
</tbody>
</table>

Running a call to the USSD Gateway product and it fails with “Prompt and Collect message build failed”

The Service Trigger is using a different Service Interface to the Service Interface that the menus were written against, even if the interfaces are the same physical interfaces.

Either rename the Service Interface for the Service Trigger or the Menus so that they are the same.

Number Normalisation is required

In the acs.conf file under acsChassis place NormalisationRule (2,E,4,06)

This strips off the 1st 4 digits of the msisdn and replaces them with 06 for all calls with an NOA 2 with any prefix. Obviously this can be changed for specific NOAs etc.
Chapter 6

About Installation and Removal

Overview

Introduction

This chapter provides details of the installation and removal process for the application.

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation and Removal Overview</td>
<td>45</td>
</tr>
<tr>
<td>CDR Loader Deployment</td>
<td>45</td>
</tr>
<tr>
<td>Post-installation Configuration</td>
<td>47</td>
</tr>
<tr>
<td>Checking Removal</td>
<td>48</td>
</tr>
</tbody>
</table>

Installation and Removal Overview

Introduction

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

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

USSD Gateway packages

An installation of the USSD Gateway includes the following packages, on the:

- SMS:
  - uisSms
  - upcSms
- SLC:
  - uisScp
  - upcScp

CDR Loader Deployment

Introduction

The USSD Interactive Service CDR Loader is installed and configured by the uisSms package. The following procedure details how to deploy the CDR Loader on a host machine other than the original machine the uisSms package was installed.
Platforms

The platform that the uisSms was originally installed on will be referred to as "platform 1" and the platform that the CDR Loader will be running on will be referred to as "platform 2" as shown below.

Files required:
1 From platform 1:
   - /IN/service_packages/UIS/bin/cdrLoader
   - /IN/service_packages/UIS/bin/cdrLoaderCron.sh
   - /IN/service_packages/UIS/bin/cmnReceiveFiles
   - /IN/service_packages/UIS/etc/cdrLoader.conf
   - /var/spool/crontabs/uis_oper

2 On platform 2:
   Assuming that the /IN and /IN/service_packages directories already exist, make the following directories:
   - /IN/service_packages/UIS
   - /IN/service_packages/UIS/bin
   - /IN/service_packages/UIS/etc
   - /IN/cdr
   - /IN/cdr/ussd
   - /IN/cdr/ussd/archives
   - /IN/cdr/ussd/failed

Procedure

Follow these steps to deploy the CDR Loader onto a different machine.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>On platform 2, make a new user &quot;uis_oper&quot;. The home directory is /IN/service_packages/UIS, and the shell is ksh.</td>
</tr>
<tr>
<td>2</td>
<td>On platform 2, add uis_oper to the allow list to run cron jobs. Edit /var/adm/cron/cron.allow and append uis_oper to the file.</td>
</tr>
<tr>
<td>3</td>
<td>On platform 2, copy the files listed above for platform 1 over to the corresponding locations onto platform 2. Ensure that the ownership and permissions are set correctly. On platform 2, duplicate the uisoperFile configuration on platform 2 from platform 1.</td>
</tr>
<tr>
<td>4</td>
<td>On platform 1, in /etc/inetd.conf there is a line starting with &quot;uisoperFile&quot;. Copy and paste the entire line into /etc/inetd.conf on platform 2 (append at the end of file). Duplicate the uisoperFile configuration on platform 2 from platform 1.</td>
</tr>
<tr>
<td>5</td>
<td>On platform 1, in /etc/inetd.conf there is a line starting with &quot;uisoperFile&quot;. Copy and paste the entire line into /etc/services on platform 2 (append at the end of file).</td>
</tr>
<tr>
<td>6</td>
<td>Restart the inet daemon by sending it the HUP signal. As root, at the prompt, find out the process ID of the inet daemon by typing the command ps -ef</td>
</tr>
</tbody>
</table>
| 7    | Send the process ID obtained above by typing `kill -HUP <process ID>`.
| 8    | Switch the SLC over to send EDR files from platform 1 to platform 2. Edit the file /IN/service_packages/UIS/bin/uisCdrPushStartup.sh At the end of the file, change `-h platform 1 to -h platform 2. |
| 9    | Result: At this point, the EDRs will be forwarded to platform 2, and the uis_oper cron job will start once a minute to check for EDRs to process. |
Chapter 6

Post-installation Configuration

Restart stats daemon

After installation of all packages, the stats daemon will need to be restarted. When the Stats daemon is started it reads the Db table smf_statistics_defn. If this table is updated, in order for these changes to take effect, the stats daemon will need to be restarted.

Follow these steps to restart the stats daemon.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Search for the stats daemon's process ID. At the prompt type: `ps -ef | grep Stats`  
Result: the result is shown:  
```
$ smf_oper 10887  1  0 09:25:42 ?  0:00
/IN/service_packages/SMS/bin/smsStatsDaemon -u
```
| 2    | Using the kill command identify the pid for the smsStatsDaemon (in this case 10887) and terminate this process. The process will then be restarted by the init daemon. |

The minimum configuration required to make a call

Here is a list of the SMS provisioning screens that need to be configured to make a call via the USSD Gateway.

- Operator
- Service IF
- Language
- Trigger Prefix

Follow these steps to set up the minimum screens required to make a call.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create a call plan by using the Call Plan Editor in ACS.</td>
</tr>
</tbody>
</table>
| 2    | Create a SAN (Service Access Number).  
To do this, bring up the ACS service screen and select Resources. Select new for service numbers, then enter a number with the associated call plan just created. |
| 3    | Create a new operator. This allows you to set up different operators against different IMSI prefixes and using different IMSI to MSISDN mapping interfaces.  
To do this, select Service>USSD Gateway>Base Config>Operator tab. |
| 4    | Create a new Service Interface. This allows you to create different service Interfaces.  
To do this, select Service>USSD Gateway>Base Config>Service IF tab. |
| 5    | Create the languages with the specific values. This unique value is viewed externally and sent to the gateway interface.  
To do this, select Service>USSD Gateway>Base Config>Language tab. |
| 6    | Create the Trigger Prefix needed. This screen allows you to create a prefix that prefixes the IMSI that can trigger a particular service field.  
To do this, select Service>USSD Gateway>Base Config>Trigger Prefix tab. |

For steps 1 and 2, consult the ACS User's Guide for further details.  
For steps 3 to 6, consult the USSD GW User's Guide for further details.
Setting up replication

Replication is a process which enables the same tables on the SMS and the SLC machines to be kept in sync. The following procedure must be followed each time packages which contain replicated tables are removed and added.

Follow these steps to create the correct config file with the replication tables for UIS, UPC on the SMS and SLC machines.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Using the SMS screen, select Operator Functions&gt;Node Management, Table Replication tab.</td>
</tr>
</tbody>
</table>
| 2    | For UIS, drag the following tables from the Available Replication Groups and drop them on the Allocated Replication Groups SLC node:  
  - UIS_OPERATOR_INFO  
  - UIS_LANGUAGE_INFO  
  - UIS_GATEWAY_INFO  
  - UIS_TRIGGER_INFO  
  - UIS_SERVICE_INTF  
  - UIS_SERVICE_TRIGGERS  
  - UIS_MENU_INFO  
  - UIS_MENU_LANGUAGE  
  - UIS_STATUS_INFO  
  - UIS_STATUS_LANGUAGE  
  - UIS_SUB_TYPE  
  - UIS_SUB_SERV_COMB  
  - UIS_IMSI_TRACE |
| 3    | For UPC, drag the following table from the Available Replication Groups and drop it on the Allocated Replication Groups SLC node:  
  - UPC_USER_SELECTION |
| 4    | Click Save. |
| 5    | Click Create Config File.  
Result: This process should indicate success. |

For more information about table replication, see SMS User's Guide.

Checking Removal

Introduction

After the un-installs have completed, it is worth double checking the /IN/service_packages/ directory to ensure the UIS, and UPC directories have gone.

Procedure

Follow these steps to check that the tables have gone.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>As sms_oper, start SQL Plus.</td>
</tr>
</tbody>
</table>
Type:
sqlplus /
select table_name from all_tables where_name like 'U%';
and make sure there are no UIS tables in the list.
NCC Glossary of Terms

AAA

ACS
Advanced Control Services configuration platform.

BCSM
Basic Call State Model - describes the basic processing steps that must be performed by a switch in order to establish and tear down a call.

CAMEL
Customized Applications for Mobile network Enhanced Logic
This is a 3GPP (Third Generation Partnership Project) initiative to extend traditional IN services found in fixed networks into mobile networks. The architecture is similar to that of traditional IN, in that the control functions and switching functions are remote. Unlike the fixed IN environment, in mobile networks the subscriber may roam into another PLMN (Public Land Mobile Network), consequently the controlling function must interact with a switching function in a foreign network. CAMEL specifies the agreed information flows that may be passed between these networks.

CC
Country Code. Prefix identifying the country for a numeric international address.

CCS
1) Charging Control Services (or Prepaid Charging) component.
2) Common Channel Signalling. A signalling system used in telephone networks that separates signalling information from user data.

CDR
Call Data Record
Note: The industry standard for CDR is EDR (Event Detail Record). Over time EDR will replace CDR in the Oracle documentation.

CPE
Control Plan Editor (previously Call Plan Editor) - software used to define the logic and data associated with a call -for example, "if the subscriber calls 0800 nnnnnn from a phone at location xxx then put the call through to bb bbb bbbb".

cron
Unix utility for scheduling tasks.

crontab
File used by cron.
CUI
Character User Interface

Diameter
A feature rich AAA protocol. Utilises SCTP and TCP transports.

DP
Detection Point

DTMF
Dual Tone Multi-Frequency - system used by touch tone telephones where one high and one low frequency, or tone, is assigned to each touch tone button on the phone.

EDR
Event Detail Record

Note: Previously CDR. The industry standard for CDR is EDR (Event Detail Record). Over time EDR will replace CDR in the NCC documentation.

FDA
First Delivery Attempt - the delivery of a short message directly to the SME rather than relaying it through the MC.

GPRS
General Packet Radio Service - employed to connect mobile cellular users to PDN (Public Data Network- for example the Internet).

GSM
Global System for Mobile communication.
It is a second generation cellular telecommunication system. Unlike first generation systems, GSM is digital and thus introduced greater enhancements such as security, capacity, quality and the ability to support integrated services.

HLR
The Home Location Register is a database within the HPLMN (Home Public Land Mobile Network). It provides routing information for MT calls and SMS. It is also responsible for the maintenance of user subscription information. This is distributed to the relevant VLR, or SGSN (Serving GPRS Support Node) through the attach process and mobility management procedures such as Location Area and Routing Area updates.

HPLMN
Home PLMN

HTML
HyperText Markup Language, a small application of SGML used on the World Wide Web.
It defines a very simple class of report-style documents, with section headings, paragraphs, lists, tables, and illustrations, with a few informational and presentational items, and some hypertext and multimedia.

**ICA**

InitiateCallAttempt. A CAMEL/INAP operation sent by the SLC to an SSP request that a voice call is started.

**IDP**

INAP message: Initial DP (Initial Detection Point)

**IMSI**

International Mobile Subscriber Identifier. A unique identifier allocated to each mobile subscriber in a GSM and UMTS network. It consists of a MCC (Mobile Country Code), a MNC (Mobile Network Code) and a MSIN (Mobile Station Identification Number).

The IMSI is returned by the HLR query (SRI-SM) when doing FDA. This tells the MSC exactly who the subscriber is that the message is to be sent to.

**IN**

Intelligent Network

**INAP**

Intelligent Network Application Part - a protocol offering real time communication between IN elements.

**Initial DP**

Initial Detection Point - INAP Operation. This is the operation that is sent when the switch reaches a trigger detection point.

**IP**

1) Internet Protocol
2) Intelligent Peripheral - This is a node in an Intelligent Network containing a Specialized Resource Function (SRF).

**IP address**

Internet Protocol Address - network address of a card on a computer

**ISDN**

Integrated Services Digital Network - set of protocols for connecting ISDN stations.

**ITU**

International Telecommunication Union

**MAP**

Mobile Application Part - a protocol which enables real time communication between nodes in a mobile cellular network. A typical usage of the protocol would be for the transfer of location information from the VLR to the HLR.
MC
Message Centre. Also known as SMSC.

MCC
Mobile Country Code. In the location information context, this is padded to three digits with leading zeros. Refer to ITU E.212 ("Land Mobile Numbering Plan") documentation for a list of codes.

MNC
Mobile Network Code. The part of an international address following the mobile country code (MCC), or at the start of a national format address. This specifies the mobile network code, that is, the operator owning the address. In the location information context, this is padded to two digits with a leading zero. Refer to ITU E.212 ("Land Mobile Numbering Plan") documentation for a list of codes.

MSC
Mobile Switching Centre. Also known as a switch.

MSIN
Mobile Station Identification Number.

MSISDN
Mobile Station ISDN number. Uniquely defines the mobile station as an ISDN terminal. It consists of three parts; the country code (CC), the national destination code (NDC) and the subscriber number (SN).

MSRN
Mobile Station Roaming Number

MT
Mobile Terminated

NOA
Nature Of Address - a classification to determine in what realm (Local, National or International) a given phone number resides, for the purposes of routing and billing.

Oracle
Oracle Corporation

PACUI
Play Announcement and Collect User Information

PLMN
Public Land Mobile Network
RRBCSM
Request Report BCSM.

SAN
Service Access Number

SCP
Service Control Point. Also known as SLC.

SCTP
Stream Control Transmission Protocol. A transport-layer protocol analogous to the TCP or User Datagram Protocol (UDP). SCTP provides some similar services as TCP (reliable, in-sequence transport of messages with congestion control) but adds high availability.

Session
Diameter exchange relating to a particular user or subscriber access to a provided service (for example, a telephone call).

SGML

SGSN
Serving GPRS Support Node

SLC
Service Logic Controller (formerly UAS).

SLEE
Service Logic Execution Environment

SME
Short Message Entity - an entity which may send or receive Short Messages. It may be located in a fixed network, a mobile, or an SMSC.

SMS
Depending on context, can be:
- Short Message Service
- Service Management System platform
- NCC Service Management System application

SMSC
Short Message Service Centre - stores and forwards a short message to the indicated destination subscriber number.
SN
Service Number

SQL
Structured Query Language - a database query language.

SRF
Specialized Resource Function - This is a node on an IN which can connect to both the SSP and the SLC and delivers additional special resources into the call, mostly related to voice data, for example play voice announcements or collect DTMF tones from the user. Can be present on an SSP or an Intelligent Peripheral (IP).

SRI
Send Routing Information - This process is used on a GSM network to interrogate the HLR for subscriber routing information.

SSF
Sub Service Field.

SSP
Service Switching Point

System Administrator
The person(s) responsible for the overall set-up and maintenance of the IN.

TCAP
Transaction Capabilities Application Part – layer in protocol stack, message protocol.

TCP
Transmission Control Protocol. This is a reliable octet streaming protocol used by the majority of applications on the Internet. It provides a connection-oriented, full-duplex, point to point service between hosts.

UIS
USSD Interactive Services

UPC
USSD Portal Components

USSD
Unstructured Supplementary Service Data - a feature in the GSM MAP protocol that can be used to provide subscriber functions such as Balance Query and Friends and Family Access.
**UUGW**

Universal USSD Gateway - used to provide data services using high performance menu-based interactions.

**VLR**

Visitor Location Register - contains all subscriber data required for call handling and mobility management for mobile subscribers currently located in the area controlled by the VLR.

**VWS**

Oracle Voucher and Wallet Server (formerly UBE).

**WAP**

Wireless Application Protocol. A standard designed to allow the content of the Internet to be viewed on the screen of a mobile device such as mobile phones, personal organisers and pagers. It also overcomes the processing limitation of such devices. The information and services available are stripped down to their basic text format.

**XML**

eXtensible Markup Language. It is designed to improve the functionality of the Web by providing more flexible and adaptable information identification.

It is called extensible because it is not a fixed format like HTML. XML is a `metalanguage` — a language for describing other languages—which lets you design your own customized markup languages for limitless different types of documents. XML can do this because it's written in SGML.
Index

#
Configuration - libupcService - 1.0- • 37
A
-a • 16
A leg • 7
AAA • 53
Abbv • 22, 23, 25
About Installation and Removal • 47
About This Document • v
ACS • 11, 43, 53
Administrative Tasks • 41
Alarms • 8
Alarms, Statistics, Reports and EDRs • 8
Audience • v
B
B leg • 7
Background Processes • 35
BCSM • 53
C
-c <if> • 14
Call back message flow • 7
Callback • 6
Callback initiation • 6
CAMEL • 53
CC • 53
CCS • 53
CDR • 29, 43, 53
CDR Loader Deployment • 47
cdrLoader • 38
Checking encoding parameters • 12
Checking procedure • 10
Checking Removal • 51
Checking the service • 43
Common Troubleshooting Procedures • 43
Components • 2
Configuration • 10, 33, 36, 38, 39
Configuration components • 9
Configuration file format • 20
Configuration Overview • 9
Configuring ACS to recognise hex digits • 11
Configuring acs.conf for the SLC • 9, 11
Configuring the SLEE.cfg • 10
Configuring the USSD Gateway Portal
Component (UPC) • 10, 17, 37
Configuring the XML Interface and Enabling Tracing • 30
Copyright • ii
CPE • 53
cron • 53
crontab • 53
CUI • 54
D
DaysOfWeek • 22
Debug • 43
Default upc.conf file • 17
defaultLanguageId • 18
destDir • 27
Diagram • 2
Diameter • 54
Document Conventions • vi
DP • 54
DTMF • 54
E
-e <opt> • 14
Editing the file • 21
Editing the SLEE.cfg file • 10
EDR • 54
EDR Section • 26
EDRs • 8
eserv.config Configuration • 10, 20, 36
eserv.config files delivered • 21
ESERV_CONFIG_FILE • 21
Example 1 • 25
Example 2 • 25
Example call flow • 5
Example call flow description • 5
Example cdrIF.cfg configuration file • 29
Example config • 26
Example SLEE.cfg file • 11
Example xmlIF.cfg • 32
Examples • 23, 24, 25
F
FDA • 54
filePrefix • 27
flushPeriod • 27
Full • 22, 23, 24
G
Gateway configuration • 9, 13, 36
GPRS • 54
GSM • 54
H
Handset integration • 3
Handset Interaction • 4
HLR • 54
HPLMN • 54
HTML • 54
I
ICA • 55
IDP • 55
IMSI • 55
Switching trace on or off, after start up • 33
System Administrator • 58
System Overview • 1

T
tagPrefix • 28
TCAP • 58
TCP • 58
tempDir • 28
The minimum configuration required to make a
call • 49
The trace file • 33
timeout • 31
timestampFormat • 29
tracingEnabled • 32
tracingPath • 31
Troubleshooting • 43
Typographical Conventions • vi

U
-u • 16
UIS • 58
UIS and UPC • 1
UPC • 58
UPC library configuration • 12
upcServiceLoader parameters • 18, 19
upcSms • 43
-user • 36
username • 39
USSD • 58
USSD Gateway packages • 47
USSD Gateway Portal Service • 3
USSD Interactive Services Gateway • 3
ussdgw • 2, 5, 22, 24, 35, 36, 43
ussdgw.sh code • 17
UssdMfileD • 36, 44
UUGW • 59

V
-v <id> • 15
VLR • 59
VWS • 59

W
-w • 17
WAP • 59
What is USSD Gateway? • 1

X
XML • 59
XML interface tracing • 33
XML script configuration • 32
xmlfile • 31
xmlIF.cfg configuration • 30