Oracle® Communications Convergent Charging Controller

USSD Gateway Technical Guide

Release 15.1

April 2025

Copyright

Contents

Chapter 6

About Installation and Removal 51

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:

- *Service Logic Execution Environment Technical Guide*
- *USSD Gateway User's Guide*

Document Conventions

Typographical Conventions

The following terms and typographical conventions are used in the Oracle Communications Convergent Charging Controller documentation.

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

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

Chapter 1

System Overview

Overview

Introduction

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

It is not intended to advise on any specific Oracle Communications Convergent Charging Controller network or service implications of the product.

In this Chapter

This chapter contains the following topics.

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
  - 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
  - with Location Capabilities Pack, session can be initiated directly back to a roaming subscriber.

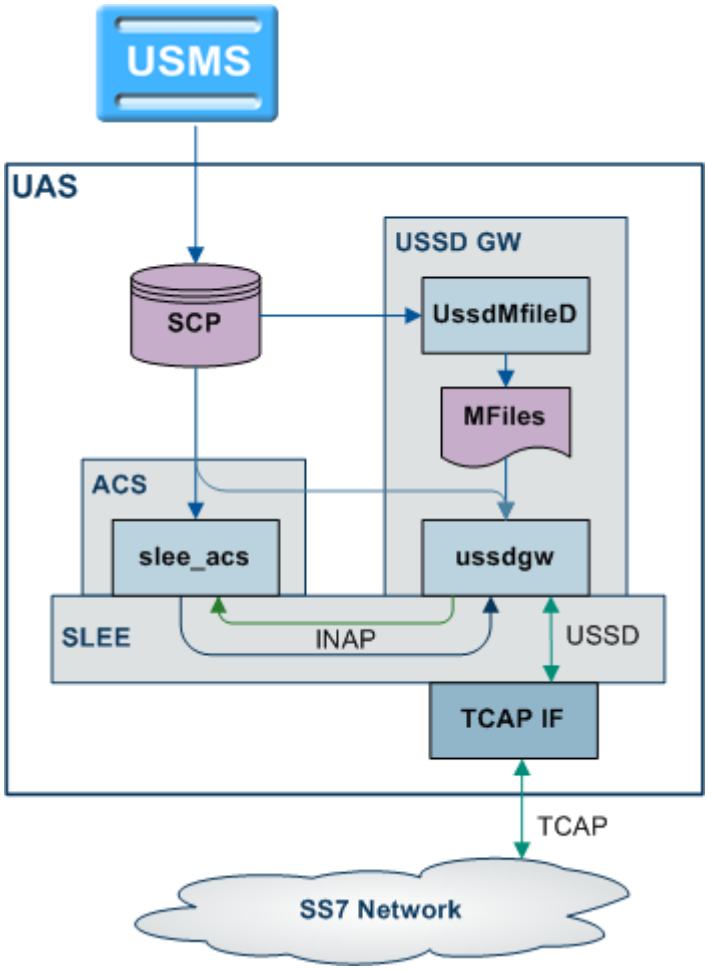
UIS and UPC

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.

Process	Role	Further information
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 39)
slee_acs	The ACS process which runs control plans.	*Advanced Control Services 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 41)

Process	Role	Further information
libupcChassisActions	libupcChassisActions provides the functions which enable the USSD GW Feature Nodes to interact with other elements in the system	*libupcChassisActions* (on page 41)
libupcMacroNodes	This slee_acs plugin provides the USSD GW macro nodes.	*libupcMacroNodes* (on page 42)

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 in 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 call back.

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).

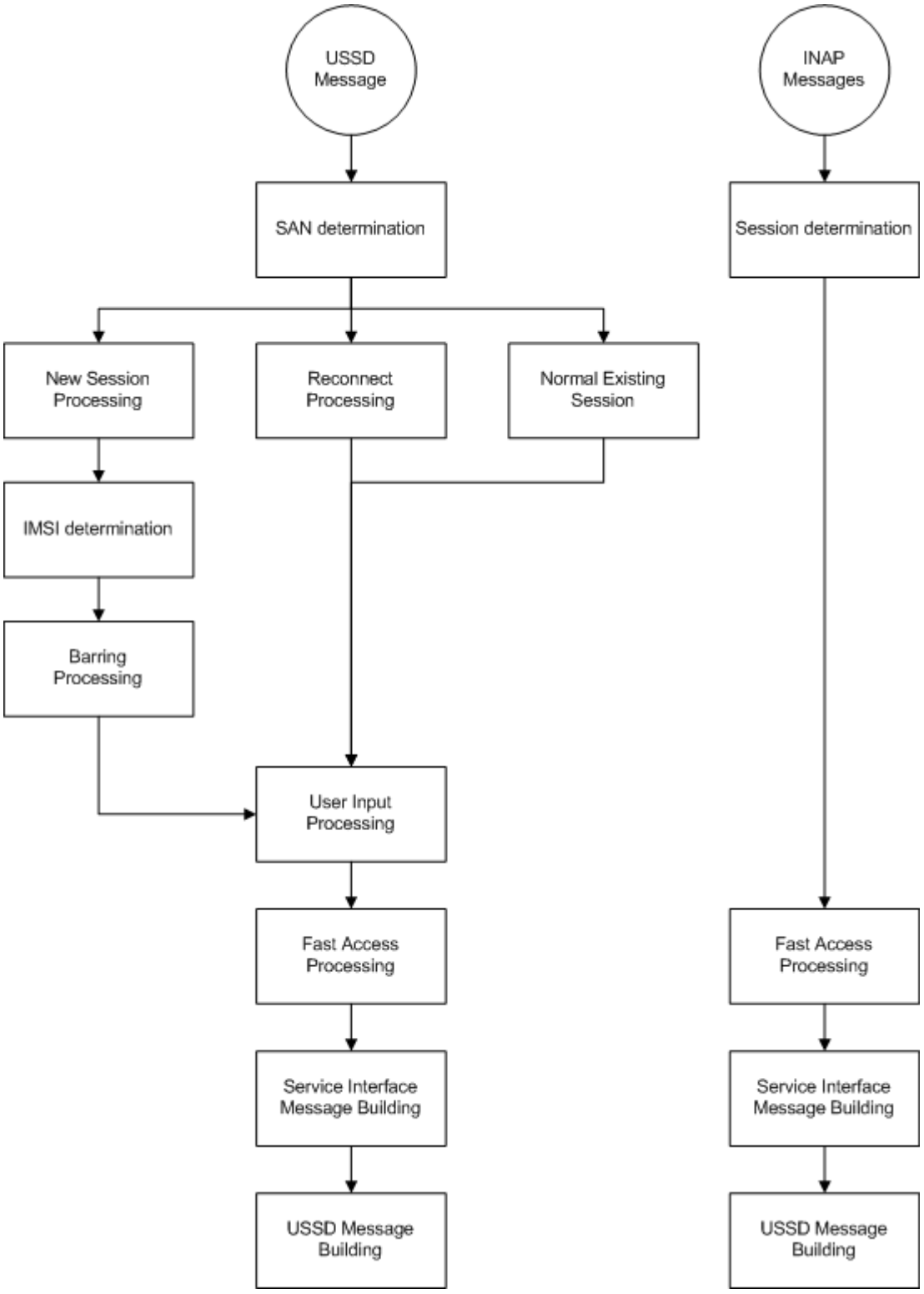
Handset integration

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

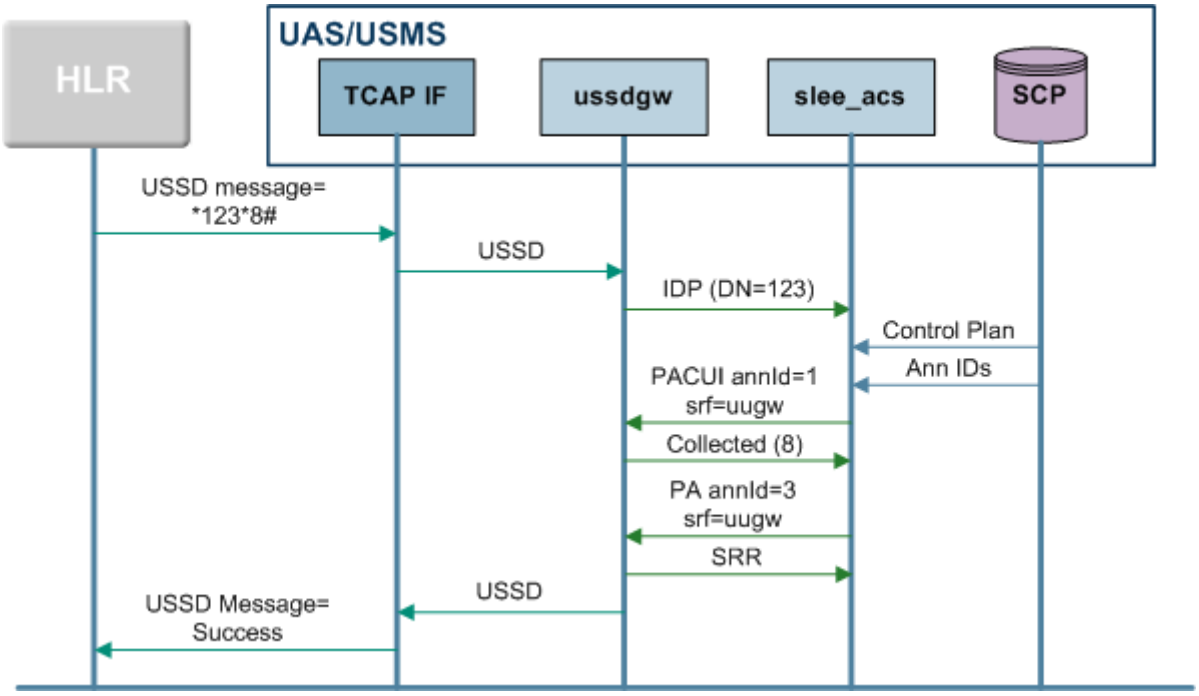
2 Typeahead, single-string commands.

In both cases, the *ussdgw* (on page 39) 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 Gateway User's Guide*.

Example call flow

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



Example call flow description

This following steps provide additional detail about the Example call flow diagram.

Flow	Description
1	The HLR sends a USSD message to the SLC, where it is picked up by the TCAP interface (usually SIGTRAN stack).
2	The TCAP interface forwards the unchanged USSD message to ussdgw across the SLEE.

Flow	Description
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 Gateway 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 *Advanced Control Services Technical Guide*.
5	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. **Warning:** 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 *srf configuration* (on page 15).
6	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.
7	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.
8	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. **Note:** If the interaction node specifies a number of repetitions of 127, ussdgw will not construct a message to be sent to the subscriber.
9	ussdgw responds to the CUI with an SRR back to slee_acs which completes the control plan.
10	The subscriber receives the USSD message from the HLR.

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 *Control Plan Editor User's Guide*.

B leg

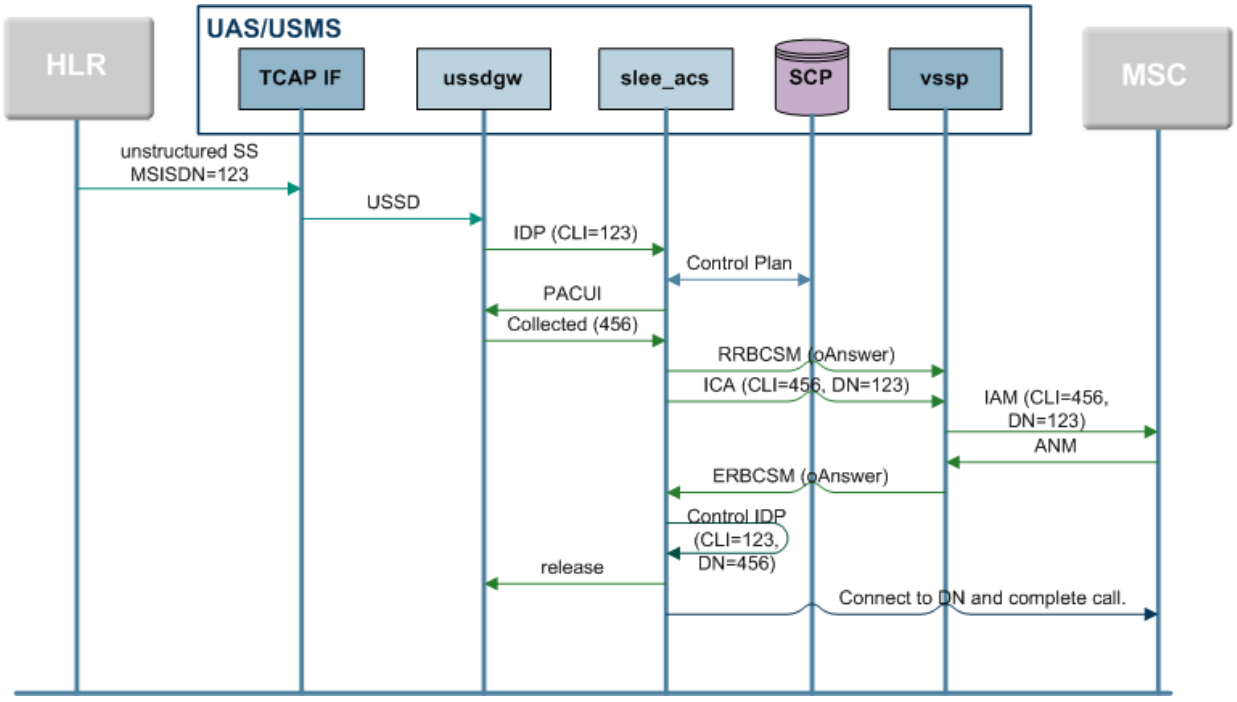
When the A party answers, the switch returns an ERBCSM (oAnswer) to the control plan and a new forked control plan starts. The new call can use any control plan functionality, including:

- 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.

Statistic	Description
UIS_1	USSD session initiation attempt – phase 1
UIS_2	USSD session initiation attempt – phase 2.
UIS_3	Successful USSD initiation attempt (InitialDP sent to a service interface)
UIS_4	Message being sent to user as a result of a PACUI INAP operation from a service interface
UIS_5	User input as a result of an active PACUI
UIS_6	Fast access attempted on USSD session initiation (that is, dial ahead digits specified)
UIS_7	Timer Expiry (Session cut off)
UIS_8	Timer Expiry (SSF)
UIS_9	Timer Expiry (overall inactivity)

Statistic	Description
UIS_10	Timer Expiry (reconnect)
UIS_11	Timer Expiry (user inactivity)
UIS_12	TC-ABORT received from network
UIS_13	TC-ABORT received from service interface
UIS_14	Gateway call limiting

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 *Event Detail Record Reference Guide*.

Chapter 2

Configuration

Overview

Introduction

This chapter explains how to configure the Oracle Communications Convergent Charging Controller application.

In this chapter

This chapter contains the following topics.

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 53).

Configuration components

USSD GW is configured by the following components:

Component	Locations	Description	Further Information
ussdgw.sh	all SLCs	ussdgw.sh sets the command line parameters which configure ussdgw.	*Gateway configuration* (on page 15)
SLEE.cfg	all SLCs	SLEE.cfg sets up SLEE interfaces and applications.	SLEE.cfg
acs.conf	all SLCs	acs.conf configures slee_acs. This includes number normalisation.	*Configuring acs.conf for the SLC* (on page 13)
SMS screens	SMF database	The service details are configured using the SMS screens.	*USSD Gateway User's Guide*

Component	Locations	Description	Further Information
upc.conf	all SLCs	If UPC is installed, upc.conf provides some of UPC's additional control plan configuration.	*Configuring the USSD Gateway Portal Component (UPC)* (on page 20)
eserv.conifg	all SLCs	eserv.config provides date formatting for outgoing messages.	*eserv.config Configuration* (on page 22)
cdrLoader.conf		cdrLoader.conf configures cdrLoader. It must be configured or cdrLoader will not start.	*Configuration* (on page 43)
cdrIF.cfg	all SLCs	Configures the EDR Interface.	Configuring 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 and 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:

Step	Action
1	cd to the following directory: **/IN/service_packages/SLEE/bin**

2	An example **slee.sh** file:
	`#!/bin/sh`
	`LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/IN/service_packages/SLEE/lib`
	`export LD_LIBRARY_PATH`
	`SHLIB_PATH=$SHLIB_PATH:/IN/service_packages/SLEE/lib`
	`export SHLIB_PATH`
	`/IN/service_packages/SLEE/bin/sleeStartup`
	`/IN/service_packages/SLEE/etc/SLEE.cfg`

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
WATCHDOGCYCLETIME=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 specifying the following configuration in the acsChassis section:

```
DialledStarEncoding C
DialledHashEncoding D
```

Note: At installation of ACS, the following the following configuration is set by default:

```
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 *Advanced Control Services Technical Guide*.

Follow these steps to ensure that ACS recognises hex digit 'C' as '*' and 'D' as '#'.

Step	Action
1	Log in to the SLC as the *acs_oper* user.
2	Edit **acs.conf** by using a text editor such as vi: **Example command:** `vi /IN/service_packages/ACS/etc/acs.conf`

3	Set these parameters to the following:
	`DialledStarEncoding C`
	`DialledHashEncoding D`
	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 *Service Logic Execution Environment 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 41)
- *libupcChassisActions* (on page 41), and
- *libupcMacroNodes* (on page 42)

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 feature node enables 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.

To use normalised calling and called party numbers in either the originating or the destination address, you must configure number normalisation in ACS. Whilst this is not configuration of the Send Buffer feature 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 srfs.
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: You start and configure ussdgw by using the ussdgw.sh shell script. For more information about ussdgw.sh, see *Startup* (on page 39).

`-i <opt>`

Syntax:	`-i <opt>`
Description:	How to populate the IMSI from a Map 2 message.
Type:	String
Optionality:	Optional (default used if not set)
Allowed:	• dest – Use the incoming MAP-OPEN destinationRefrence for the IMSI. • orig – Use the incoming MAP-OPEN originationRefrence for the IMSI. • none – Do not populate the IMSI. • The numeric IMSI value – A string of digits, which is the actual value of the IMSI, to put in the IMSI.
Default:	None
Notes:	
Example:	`-i orig`

`-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:
Default: Global GW config
Notes:
Example:

`-c <if>`

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

`-s <opt>`

Syntax `-s <opt>`
Description: How to populate the MSISDN from a MAP 2 message.
Type: String
Optionality: Optional
Allowed:
- msisdn – Use the MSISDN from the body of the operation.
- msisdnref – Use the incoming msisdnReference for the MSISDN.
- orig – Use the incoming originatingReference for the MSISDN.
- dest – Use the destinationReference for the MSISDN.
- none – Do not populate the MSISDN.

Default: None
Example: `-s msisdnref`

`-o <opt>`

Syntax: `-o <opt>`
Description: How to populate the MSISDN in the IDP.
Type: String
Optionality:
Allowed:
- imsi – Use the IMSI from the incoming Map1BeginSubscriberActivity for the MSISDN in the IDP.
- oen – Use the OriginatingEntityNumber from the incoming Map1BeginSubscriberActivity for the MSISDN in the IDP.
- none – Do not populate the MSISDN in the IDP.

Default:

Notes:
Example: `-o oen`

`-e <opt>`

Syntax: `-e <opt>`
Description: How to populate the IMSI from a MAP 1 message.
Type: String
Optionality:
Allowed:

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

Default:
Notes:
Example: `-e imsi`

`-v <id>`

Syntax: `-v <id>`
Description: The VLR announcement set id.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: - (disabled by default)
Notes:
Example:

`-m <max>`

Syntax: `-m <max>`
Description: The maximum number of concurrent calls allowed.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: - (unlimited 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: `-r send_PA_on_Rel_MAP`

`-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:

`-u`

Syntax: `-u`
Description: Send Unicode Characters.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed:
Default: - (disabled by default)
Warning: Your db character set must be UTF8 to send Unicode.
Example:

`-d`

Syntax: `-d`
Description: Used after -u option to escape the encoding of USSD message in UCS-2.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed:
Default: - (disabled by default)

Notes: This command line parameter informs USSD that the message is plain English text and don't need UCS-2 encoding.

After setting this command line parameter, you can send message up to 150 characters (only English).

Warning: Use only after -u command-line parameter.

Example:

`-z <str>`

Syntax: `-z <str>`

Description: The string to change "+"s in a number from the handset to.

Type: String of zero or more digits

Optionality: Optional (no replacements are made if not set).

Allowed:

Default: None

Notes:

Example: `-z 00`

This rule would change an incoming number of +641234567890 to 00641234567890.

`-w`

Syntax: `-w`

Description: Strips leading and trailing white space from the typeahead text returned in PACUI responses.

Type:

Optionality: Optional (no white space removal actions).

Allowed:

Default: None

Notes: This rule would change incoming typeahead text of:

*103*1239*1239 # to

*103*1239*1239#

Example: `-w`

`-g <opt>`

Syntax: `-g <opt>`

Description: Set a data coding group.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

- 4 (for iOS devices)
- 5 (for Android devices)

Default: 5

Notes:

Example: `-g 4`

ussdgw.sh code

Here is the startup script code.

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

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.

```
# Service loader configuration
upcServiceLoader

    # 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!
:
```

upcServiceLoader 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 "upcServiceLoader".

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: `defaultLanguageId` *<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: `noMsisdnReleaseCause` *<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: `smscInterfaceName` *<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:	`smscFromAddress` <*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:	
Default:	
Notes:	
Example:	

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.

Step	Action
1	Log in as upc_oper # su - upc_oper
2	Edit upc.conf: `$ vi /IN/service_packages/UPC/etc /upc.conf`
3	Add `noMsisdnReleaseCause` <*value*> within the *upcServiceLoader* (on page 20) section. Note that indentation may be added for human-readibility.
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 Oracle Communications Convergent Charging Controller applications read their configuration. Each Convergent Charging Controller machine (SMS, SLC, and VWS) has its own version of this configuration file, containing configuration relevant to that machine. The **eserv.config** file contains different sections; each application reads the sections of the file that contains data relevant to it.

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

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

Configuration File Format

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

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

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

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

or

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

or

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

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 causes file errors when the application tries to read the configuration file.

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

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 39) 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:

Default: As set by *Full* (on page 25) and *Abbv* (on page 25) 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 39) 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 = {
        Full = [ "Minggu", "Senin", "Selasa", "Rabu", "Kamis",
        "Jumat", "Sabtu" ]
        Abbv = [ "Min", "Sen", "Sel", "Rab", "Kam", "Jum", "Sab" ]
    }
}
```

Language

Syntax: For an example of how to use this parameter, see *DaysOfWeek* (on page 24).

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 27).

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 27).

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 27).

Months

Syntax:

```
Months = {
     <lang> = {
          Full = [ <config> ]
          Abbv = [ <config> ]
     }
     [...]
 }
```

Description: The names of the months of the year in various configured languages used in outgoing USSD messages.

Type: Array

Optionality: Optional (defaults used if not set).

Allowed:

Default: As set by *Full* (on page 26) and *Abbv* (on page 27) parameters.

Notes: There must an entry for each of all the 12 months of the year or an alarm will be logged when *ussdgw* (on page 39) starts up and the default language will be used.

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

Example:

```
Months = {
     English = {
          Full = [ "January", "February", "March", "April", "May",
          "June", "July", "August", "September", "October",
          "November", "December" ]
          Abbv = [ "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
          "Aug", "Sep", "Oct", "Nov", "Dec" ]
     }
     Bahasa = {
          Full = [ "Januari", "Februari", "Maret", "April", "Mei",
          "Juni", "Juli", "Agustus", "September", "Oktober",
          "November", "Desember" ]
          Abbv = [ "Jan", "Feb", "Mar", "Apr", "Mei", "Jun", "Jul",
          "Agu", "Sep", "Okt", "Nov", "Des" ]
     }
 }
```

Language

Syntax: For an example of how to use this parameter, see *Months* (on page 26).

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 27).

`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:	
Default:	January, February, March, April, May, June, July, August, September, October, November, December
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 27).

`Abbv`

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:	
Default:	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 27).

Examples

Example 1

This text shows an example of 1 Language:

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

```
DateAndTime = {
    DaysOfWeek = {
        Bahasa = {
            Full = [ "Minggu", "Senin", "Selasa", "Rabu",
                     "Kamis", "Jumat", "Sabtu" ]
            Abbv = [ "Min", "Sen", "Sel", "Rab", "Kam", "Jum", "Sab" ]
        }
    }
    Months = {
        Bahasa = {
            Full = [ "Januari", "Februari", "Maret", "April",
                     "Mei", "Juni", "Juli", "Agustus",
                     "September", "Oktober", "November", "Desember" ]
            Abbv = [ "Jan", "Feb", "Mar", "Apr", "Mei", "Jun",
```

```
                    "Jul", "Agu", "Sep", "Okt", "Nov", "Des" ]
        }
    }
}
```

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 = {
            Full = [ "Minggu", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu" ]
            Abbv = [ "Min", "Sen", "Sel", "Rab", "Kam", "Jum", "Sab" ]
        }
        Japanese = {
            Full = [ "Nichiyoobi", "Getsuyoobi", "Kayoobi", "Suiyoobi", "Mokuyoobi",
            "Kin'yoobi", "Doyoobi" ]
            Abbv = [ "Nic", "Get", "Kay", "Sui", "Mok", "Kin", "Doy" ]
        }
        Polish = {
            Full = [ "Niedziela", "Poniedziałek", "Wtorek", "Środa", "Czwartek",
            "Piątek", "Sobota" ]
            Abbv = [ "Nie", "Pon", "Wto", "Śro", "Czw", "Pią", "Sob" ]
        }
    }

    Months = {
        English = {
            Full = [ "January", "February", "March", "April", "May", "June", "July",
            "August", "September", "October", "November", "December" ]
            Abbv = [ "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep",
            "Oct", "Nov", "Dec" ]
        }
        Bahasa = {
            Full = [ "Januari", "Februari", "Maret", "April", "Mei", "Juni", "Juli",
            "Agustus", "September", "Oktober", "November", "Desember" ]
            Abbv = [ "Jan", "Feb", "Mar", "Apr", "Mei", "Jun", "Jul", "Agu", "Sep",
            "Okt", "Nov", "Des" ]
        }
        Japanese = {
            Full = [ "Ichigatsu", "Nigatsu", "Sangatsu", "Shigatsu", "Gogatsu",
            "Rokugatsu", "Shichigatsu", "Hachigatsu", "Kugatsu", "Juugatsu",
            "Juuichigatsu", "Juunigatsu" ]
            Abbv = [ "Ichi", "Ni", "San", "Shi", "Go", "Roku", "Shichi", "Hachi",
            "Ku", "Juu", "Juuichi", "Juuni" ]
        }
        Polish = {
            Full = [ "Styczeń", "Luty", "Marzec", "Kwiecień", "Maj", "Czerwiec",
            "Lipiec", "Sierpień", "Wrzesień", "Październik", "Listopad", "Grudzień" ]
            Abbv = [ "Sty", "Lut", "Mar", "Kwi", "Maj", "Cze", "Lip", "Sie", "Wrz",
            "Paź", "Lis", "Gru" ]
        }
    }
}
```

UIS Section

Configuring USSD Gateway to Prevent SLEE Overload

USSD Gateway marks itself as overloaded and throttles all new transactions for the period of time configured in the `backoffPeriodMilliseconds` parameter whenever it fails to create a SLEE dialog due to overload.

You configure this option in the UIS section of the **eserv.config** file. For example:

```
UIS = {
    ...
    backoffPeriodMilliseconds = 1000
    ...
}
```

where the `backoffPeriodMilliseconds` value specifies the period of time in milliseconds during which requests are rejected.

Configuring USSD Gateway to Add Spaces between Elementary Messages

If USSD menu is dynamically formed with multiple elementary messages, then a space is added between each elementary message text. Adding of this space is controlled by `addSpaceBetweenElementaryMsgs` parameter present in the UIS section of the eserv.config file. For example:

```
UIS = {
      ...
      addSpaceBetweenElementaryMsgs = 0
      ...
}
```

The allowed parameter values are:

- 0: Don't add space between elementary messages
- 1: Add space between elementary messages

By default, the value is set as 1.

Parameters

These are the UIS parameters:

`msisdnMappingOption`

Syntax:	`msisdnMappingOption = "<value>"`
Description:	It allows to configure the normalization of MSISDN and mapping of noa field.
Type:	String
Optionality:	Optional
Allowed:	I = Normalise the number to International N = Normalise the number to National n = Map the MAP NOA to the INAP NOA equivalent only u = Set the NOA to UNKNOWN (2)
Default:	n
Notes:	
Example:	`msisdnMappingOption = "I"`

InternationalEscCodes

Syntax: `InternationalEscCodes = "<value>"`
Description: Numbers to be skipped from subscriber number before normalization.
Type: String
Optionality: Optional
Allowed: Valid Integer
Default: “” (Empty String)
Notes:
Example: `InternationalEscCodes = "31"`

countryCode

Syntax: `countryCode = "<value>"`
Description: Number to be skipped from subscriber number before normalization.
Type: String
Optionality: Optional
Allowed: Valid Country Code
Default: “62”
Notes:
Example: `countryCode = "55"`

backoffPeriodMilliseconds

Syntax: `backoffPeriodMilliseconds = "<value>"`
Description: ussdgw process marks itself as overloaded and throttle all new transactions for the configured `backoffPeriodMilliseconds` whenever it fails to create a dialog due to overload.
Type: Unsigned Integer
Optionality: Optional
Allowed: 1 to 4294967295
Default: 1000
Notes:
Example: `backoffPeriodMilliseconds = "500"`

addSpaceBetweenElementaryMsgs

Syntax: `addSpaceBetweenElementaryMsgs = 0 | 1`
Description: To decide whether to add spaces between elementary messages.
Type: Boolean
Optionality: Optional
Allowed: 0 | 1
Default: 1
Notes:
Example: `addSpaceBetweenElementaryMsgs = 1`

releaseCallHandlingModeOnSessionTimeout

Syntax:	releaseCallHandlingModeOnSessionTimeout = "<value>"
Description:	Used to handle call release on a session timeout.
Type:	Character
Optionality:	Optional
Allowed:	'0' / '1' / '2' '0': Existing implementation of sending PUSSR (Process Unstructured Supplementary Service Request) on PA/PAC request timeout. '1': Sending Uss-Notify on PA/PAC request timeout. '2': Sending Map error on PA/PAC request timeout.
Default:	'0'
Notes:	
Example:	releaseCallHandlingModeOnSessionTimeout = '0'

paTimeoutErrorCode

Syntax:	paTimeoutErrorCode = "<value>"
Description:	This is the value which will be sent in the errorcode field in the mapError message on PA message timeout and when releaseCallHandlingModeOnSessionTimeout is set to '2'.
Type:	Integer
Optionality:	Optional
Allowed:	Integer
Default:	50
Notes:	
Example:	paTimeoutErrorCode = 50

pacTimeoutErrorCode

Syntax:	pacTimeoutErrorCode = "<value>"
Description:	This is the value which will be sent in the errorcode field in the mapError message on PAC message timeout and when releaseCallHandlingModeOnSessionTimeout is set to '2'.
Type:	Integer
Optionality:	Optional
Allowed:	Integer
Default:	51
Notes:	
Example:	pacTimeoutErrorCode = 51

useOriginalInvokeIdInReleaseCall

Syntax:	useOriginalInvokeIdInReleaseCall = 0 | 1
Description:	Will specify what invokeid value need to be sent in the mapError message. 0: send current invoke id i.e. invokeid from previous request. 1: send original invoke id which is received in PUSSR.
Type:	Boolean
Optionality:	Optional

Allowed:	0 | 1
Default:	0
Notes:	
Example:	`useOriginalInvokeIdInReleaseCall = 1`

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:

```
UIS = {
    EDR = {
        flushPeriod = 1800    # 30 minutes
        filePrefix = "UIS"
        tagPrefix = "UIS"
        destDir = "/IN/cdr/UIS/closed"
        tempDir = "/IN/cdr/UIS/current"
        maxNum = 10000
        timestampFormat = "%Y-%m-%d %T[usec:6]"
        sessionIdMask = "3fffffff"
    }
}
```

Parameters

These are the EDR parameters:

`destDir`

Syntax:	`destDir = "<directory>"`
Description:	Base file store directory for completed EDR files.
Type:	String
Optionality:	Optional (default used if not set).
Allowed:	
Default:	"/IN/service_packages/UIS/edr/closed"
Notes:	**Warning:** The directory specified here must exists and have the correct permissions for the user that executes the ussdgw process.
Example:	`destDir = "/IN/service_packages/UIS/edr/closed"`

`filePrefix`

Syntax:	`filePrefix = "<>"`
Description:	Base filename used to create EDR log files.
Type:	String
Optionality:	Optional (default used if not set).
Allowed:	

Default:	"" - null value
Notes:	Could be used to easily indicate which process the EDR file was generated by.
Example:	`filePrefix = "UIS"` would give a file name of: "UIS" + pid + "YYYYMMDDHHMMSS.cdr"

flushPeriod

Syntax:	`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:	0 - A value of zero indicates that no EDR files will be generated.
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`

sessionIdMask

Syntax:	`sessionIdMask = <value>`
Description:	sessionIdMask is used with the ACS Call ID to create the value for the session ID by performing bitwise AND operation.
Type:	String
Optionality:	Optional
Allowed:	32 bit hexadecimal value
Default:	0xffffffff
Notes:	
Example:	`sessionIdMask = "3fffffff"`

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: `tempDir = "/IN/service_packages/UIS/edr/current"`

timestampFormat

Syntax: `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 **man strftime.**

Default: `"%Y-%m-%d %T"`

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

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

Example: `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 [usec 4], same timestamp will appear as:

```
2010-02-18 03:59:09.5997
```

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 192.0.2.64`

`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:
Allowed: true, false
Default: true
Notes:
Example: `keepalive true`

Example xmlIF.cfg

Here is an example xmlIF.cfg file.

```
ip 192.0.2.64
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 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:

`tracingPath`	The directory where the trace can be created.
`tracingEnabled`	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.

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

```
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.

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

The trace file

A new trace file is created each day at midnight. The trace file name has the following format:

xmlTrace_*date*.log

where *date* is the date the file is created.

For example:

xmlTrace_20030622.log

Chapter 3

Background Processes

Overview

Introduction

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

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

In this chapter

This chapter contains the following topics.

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 15) and *eserv.config Configuration* (on page 22).

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 39).

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:

```
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:	

`-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 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 20).

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 the SMF database.

cdrLoader is required to view EDRs in the CDR Viewer screen.

Location

cdrLoader is located on SMS nodes.

Startup

cdrLoader runs, every minute by default, in the crontab for the *acs_oper* user. It is run by the following script:

/IN/service_packages/UIS/bin/cdrLoaderCron.sh

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

Configuration

cdrLoader is configured in the **cdrLoader.conf** file, located in the following directory:

/IN/service_packages/UIS/etc/cdrLoader

cdrLoader will start only if **cdrLoader.conf** can be found at this location.

For connections to a local or a remote database through the Oracle wallet secure external password store, set the following parameter:

```
nsname=/@connection_string
```

where *connection_string* is the alias of the credentials in the external password store. *connection_string* can be either a TNS name or a service name specified in **tnsnames.ora**. For remote connections, the name must reference the remote database.

For connections to a local database by specifying user credentials, set the following parameters:

```
username=user
password=password
```

where *user* is the name of the user authorized to log in to the Oracle database on the local system and *password* is the password for the specified user. To use the default login (*/*), do not specify any parameters in the **cdrLoader.conf** file. The **cdrLoader.conf** file must exist, but it can be empty.

For connections to a remote database without using the external password store, set the following parameters:

```
username=user
password=password
nsname=nsname
```

where:

- *user* and *password* are the user credentials for logging in to the Oracle database on the remote system.
- *nsname* is the name of the remote database.

Chapter 4

Administrative Tasks

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

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.

Step	Action
1	As the *acs_oper* user, enter the following command: `/IN/service_packages/SLEE/bin/slee.sh` **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, for example by specifying: `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.

Step	Action
1	As the *acs_oper* user, enter the following command: `/IN/service_packages/SLEE/bin/stop.sh`

Step	Action
2	After stopping the service, run the clean process by entering: `./clean`

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 by using the 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.

Chapter 5

Troubleshooting

Overview

Introduction

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

In this chapter

This chapter contains the following topics.

Common Troubleshooting Procedures

Introduction

Refer to *System Administrator's Guide* for troubleshooting procedures common to all Convergent Charging Controller 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

The following table provides a list of possible problems and the course of action required to fix each problem.

Problem	Remedy

The service does not appear to be running as expected	Check that the service is actually running. If it is, using the Unix ps command, you should get a response similar to the following:
	`# ps -fu acs_oper`
	`acs_oper 1975 1 0 11:38:07 pts/7 0:00 ./ussdgw --oracle-login / --cdr-interface cdrIF`
	The main thing to note here is that the *ussdgw* (on page 39) process is running. If it is not running, then the SLEE has not been started.

EDRs are not being written	Check that the CDR flag in the Gateway configuration has been set, as described in the *USSD GW User's Guide*.

Problem	Remedy
	Check that the user who runs the SLEE (acs_oper) has permission to write to the **/IN/cdr/temp and /IN/cdr/ussd** directories. Once the permissions are changed, the cdrIF will start up automatically and EDRs will be written to a file in the **/IN/cdr/temp** directory. When the files in this directory reach the defined size, set in **cdrIf.cfg**, they will be output into the **/IN/cdr/ussd** directory.
Alarms are not being logged	To check for alarms, use: `tail -f /var/adm/syslog/syslog.log`. If alarms are not being logged, then the SMS alarm subsystem has not been properly installed. Contact your system administrator.

When starting the SLEE, the following error appears:
`SLEE Exception (1005) in sleeUnixSemaphore.cc at 114 by process id 2537`
	You need to increase the number of semaphores and rebuild the kernel.
	`SEMMNS`
	`SEMMNI`
	`SEMMSL`
	Then rebuild the kernel using the sam command, and reboot the machine.

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`

Problem	Remedy

Running a call to the USSD Gateway product and it fails with “Service Trigger Undetermined”	One of the *UssdMfileD* (on page 40) 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.

When removing the upcSms package the following is displayed:
`./upcSms.unconf.sh: ./checkCanUnconfigure.sh : not found`
`* The following macro nodes are still in use by one or more call plans ...`
`* Version Branching`
`* User Selection`
`* Language Setting`
`* User Input`
`* Send Buffer`
`* Call plans using these nodes should be exported first. Abort uninstallation ? [y,n,?]`
	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. 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.

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. This can be changed for specific NOAs etc.

Chapter 6

About Installation and Removal

Overview

Introduction

This chapter provides information about the installed components for the Oracle Communications Convergent Charging Controller application described in this guide. It also lists the files installed by the application that you can check for, to ensure that the application installed successfully.

In this Chapter

This chapter contains the following topics.

Installation and Removal Overview

Introduction

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

- Convergent Charging Controller system requirements
- Pre-installation tasks
- Installing and removing Convergent Charging Controller 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.

Step	Action
1	On platform 2, make a new user "uis_oper". The home directory is /IN/service_packages/UIS, and the shell is ksh.
2	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.
3	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.
4	On platform 2, duplicate the uisoperFile configuration on platform 2 from platform 1.
5	On platform 1, in /etc/inetd.conf there is a line starting with "uisoperFile". Copy and paste the entire line into /etc/inetd.conf on platform 2 (append at the end of file).
6	Duplicate the uisoperFile configuration on platform 2 from platform 1.
7	On platform 1, in /etc/services there is a line starting with "uisoperFile". Copy and paste the entire line into /etc/services on platform 2 (append at the end of file).
8	Restart the inet daemon by sending it the HUP signal.
9	As root, at the prompt, find out the process ID of the inet daemon by typing the command `ps -ef |grep inetd`.
10	Send the process ID obtained above by typing `kill -HUP <`*process ID*`>`.
11	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`. **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.

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.

Step	Action

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.

Step	Action
1	Create a call plan by using the Call Plan Editor in ACS.
2	Create a SAN (Service Access Number). To do this, bring up the ACS service screen and select **Resourses**. 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 *Advanced Control Services User's Guide* for further details.
For steps 3 to 6, consult the *USSD Gateway 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.

Step	Action
1	Using the SMS screen, select **Operator Functions**>**Node Management**, **Table Replication** tab.
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 *Service Management System 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.

Step	Action
1	As sms_oper, start SQL Plus.

Step	Action

2	Type:
	`sqlplus /`
	`select table_name from all_tables where_name like 'U%';`
	and make sure there are no UIS tables in the list.