Oracle® Communications Convergent Charging Controller

SMS Center Technical Guide

Release 15.0.0

October 2023

Copyright

Contents

About This Document

Scope

The scope of this document includes all functionality a user must know in order to effectively install and configure the SMS Interface.

The purpose of this document is to describe how to use the SMS Interface on an intelligent Network platform.

The document is not intended to detail the technical design of the SMS Interface or to advise on the network implications of operating the SMS Interface.

Audience

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

Prerequisites

Although there are no prerequisites for using this guide familiarity with Intelligent Network architectures would be an advantage.

As well as an understanding of the Short Message Service center EMI-UCP Interface specification Version 3.5.

Related Documents

The following documents are related to this document:

- [1] EMI – UCP Interface Specification 3.5

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.

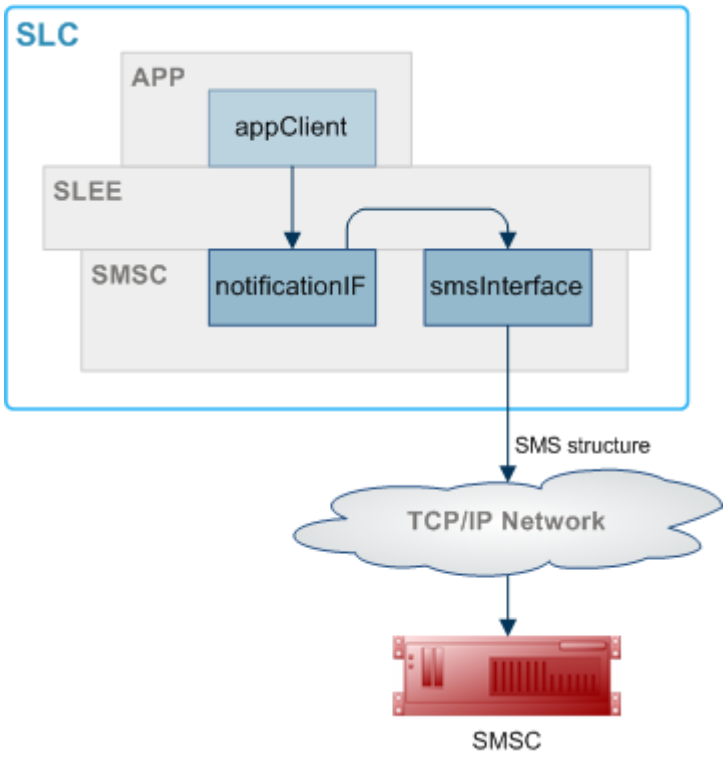
SMS Interface System Environment

What is the SMS interface?

SLEE applications can use the smsInterface to send messages to an SMS center. Messages use the UCP protocol defined in *EMI-UCP Interface Specification*.

Architectural overview

The following diagram shows the smsInterface and sub-system components that surround it.



Sub-system components

This table provides notes about components surrounding the smsInterface.

Component	Notes
SLEE	The service logic execution environment. The SLEE: • Starts and stops the smsInterface • Directs communication to the smsInterface from other Convergent Charging Controller applications. The SLEE must be installed before the smsInterface.
smsclF.cfg file	A configuration file, **smsclF.cfg.example**, is supplied in the form of an example. You can find it at **/IN/service_packages/SLEE/etc** If you use this file you must rename it to: **smsclF.cfg**
Log file	The log file, **smsclF.log**, is created at run time. You can find it at **/IN/service_packages/SLEE/tmp**
Trace log	The trace log file is created at run time. You can find it either at **/IN/service_packages/UCP/tmp** or, if the UCP directory does not exist, at **/tmp**
Prefix file	The prefix file, **smscPrefix.cfg**, is created as part of the user's configuration process. You can find it at **/IN/service_packages/SLEE/etc**
SMS message structures	SMS message structures supported are: • Submit Short Message Operation(51) • Session Management Operation(60) See *EMI-UCP Interface Specification*.

smsInterface Connection Management

Purpose

The smsInterface can queue and manage the sending of SMS messages to one or more SMSCs.

Message queuing

The message events are held in one of two queues.

Queue	Description
Live	This queue contains those messages currently awaiting acknowledgment, up to a maximum of 100 messages. Messages are only placed on this queue if there is a live connection to an SMSC and if there is room for more messages. Messages are removed from the live queue when they have been acknowledged or if they time out.
Holding	If all 100 'slots' in the live queue are occupied by messages that have not completed or if there is no live connection the messages are placed in this queue. This queue has a configurable (*queueMaximum* (on page 14)) size. A message may be removed from the holding queue if the message is older than *msgLife* (on page 13). Any remaining messages will be re-sent upon successful re-connection to a SMSC. This occurs as part of the close connection process called whenever messages are being transmitted to the SMSC and a failure is detected.

Connection process

The IP addresses of two SMSCs can be configured.

The IP addresses are to be read from a configuration file.

During the connection process an attempt is first made to connect to the current (or initial) SMSC. Upon program startup the Primary machine will be the first one that a connection attempt is made to.

Should this fail then a count of sequential failures is made. If this number is greater than the configuration file parameter (retry number) then a connection is attempted to the other SMSC (there is a timed delay set by the configuration file parameter (retry interval) between each new connection attempt).

Should the attempt to this second SMSC also fail (retry number) times a configured delay of (fail over interval) is made before a re-connect attempt is made to the OTHER SMSC.

A user signal (USR2) can be sent that will trigger a change back to the primary IP. This will call the close connection process resulting in all those messages that are in the 100 element live queue being resent when a new connection is achieved.

Connection failure

If a connection fails, remaining messages are moved from the live queue to the holding queue and the connection is terminated as part of the close connection / connect process.

The connection will be considered failed if any of the following conditions are met:

- Acknowledgment timeout (acknowledge timeout) - No response from the SMSC.
- Idle timeout - No communication for period of time defined by *idleTimeout* (on page 12).
- SMS negative acknowledgment error - Failure message is received from the SMSC.
- Read / Write failure of connection.

When messages are placed in the live queue they are removed from the holding queue.

Login event

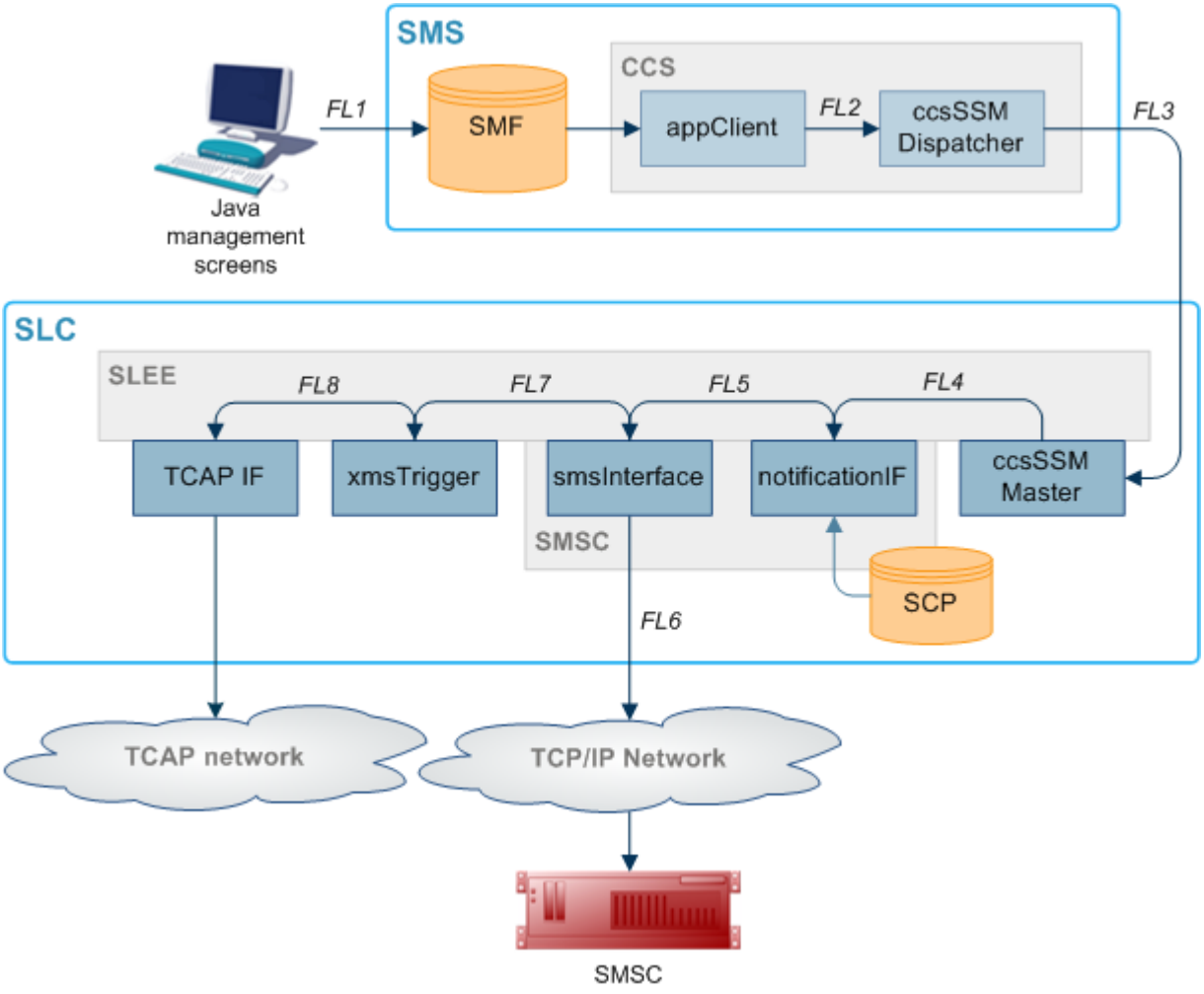
A close connection / re-connect may also arise from a failure of the login message.

The login message is sent after a successful connection but before any attempt to send SMS messages is made. The login message expects a result back from the SMSC. If it does not get one within the configuration file parameter (login response time) or an error is returned then a close connection process is initiated followed by the reconnection process.

Notification Interface

Architectural overview

The following diagram shows the notificationIF and sub-system components that surround it It also identifies notificationIF message flows.



Message flows

The Notification interface uses the message flows identified in the diagram as FL1 through FL 8.

Flow Label	Description
FL1	Java management screens access the SMF database to edit the list of SMS templates.
FL2	A client application sends a request, via a FIFO, to the ccsSSMDispatcher running on the SMS. An example of a client application would be a smsTrigDaemon.
FL3	The ccsSSMDispatcher forwards requests to the ccsSSMMaster running on the SLC platform. Requests are forwarded through a TCP/IP socket.
FL4	ccsSSMMaster constructs a SLEE notification event with: • Application name • Notification type • Language • Destination MSISDN • Optional variable parts SLEE notification events are sent to the Notification interface.
FL5	notificationIF is able to forward requests to smsInterface.
FL6	The smsInterface sends requests it receives from the notificationIF to the SMSC. Requests are forwarded through a TCP/IP network.
FL7	notificationIF is also able to forward requests to Messaging Manager's xmsTrigger process.
FL8	xmsTrigger sends requests it receives from the notificationIF directly to the destination. It uses the TCAP FDA interface.

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.

Configuring the Environment

Startup configuration

notificationIF and smsInterface are started by shell scripts. They can be used to set environmental parameters. For more information about smscIF.sh, see *SLEE.cfg* (on page 8).

SMSC supports the following environmental parameters:

`LOG_FILE`

Syntax:	`LOG_FILE=`*`PathFile`*
Description:	The name of the file to which alarms raised by the smsInterface will be logged.
Type:	String
Optionality:	Optional (default used if not set).
Allowed:	
Default:	
Notes:	For more information about SMSC alarms, see System Alarms.
Example:	`LOG_FILE=/IN/service_packages/SLEE/tmp/smscIF.log` `LOG_FILE=/IN/service_packages/SLEE/tmp/notificationIF.log`

`CONF_FILE`

Syntax:	`CONF_FILE=`*`PathFile`*
Description:	The directory smsInterface will look for smscIF.sh configuration file in.
Type:	String
Optionality:	Optional (default used if not set).
Allowed:	
Default:	/IN/service_packages/SLEE/etc/smscIF.sh
Notes:	

Example: `CONF_FILE=/IN/service_packages/SLEE/etc/smscIF.cfg`

Location of eserv.config

By default, notificationIF will read its configuration from the notificationIF 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:	

SLEE.cfg

The SLEE is responsible for starting and stopping the smsInterface and notificationIF.

The SLEE also directs all communication to the smsInterface from other Oracle components.

A row in the **SLEE.cfg** file tells the SLEE which executable to run and gives it an identifier for other applications. For more information about this line, see *Startup* (on page 22).

Multiple instance configuration

Multiple occurrences of the *smsInterface* (on page 22) may be configured within the **SLEE.cfg** file.

This can allow SMS messages to be sent to more than one SMSC by creating multiple **smscIF.cfg** files (one for each unique instance of the smsInterface) and within these configuration files, defining different smsInterface IPs.

Each instance defined within the **SLEE.cfg** file must have a unique handle. See Single Instance Configuration above.

Thus two entries within the **SLEE.cfg** file might be of the form:

```
INTERFACE=smscIF smscIF.sh /IN/service_packages/SLEE/bin UDG
INTERFACE=smscIF2 smscIF2.sh /IN/service_packages/SLEE/bin UDG
```

Note: There must always be one instance with the handle smscIF.

For more information about **SLEE.cfg** and INTERFACE entries, see *SLEE Technical Guide*.

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" ]
}
```

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.

Example eserv.config file

To see an example of a configured **eserv.config** file, see *Example notificationIF section* (on page 21).

Configuring the smscIF.cfg

Overview

The *smsInterface* (on page 22) needs to be set up to connect to the required SMSC. You do this by setting the values of parameters in the **smscIF.cfg** configuration file.

Configuration file structure

The structure of smscIF.cfg is summarized below.

- The configuration file consists of a number of lines, each of which defines a parameter name and associated value or values.
- Each line can contain characters totaling no more than 1024 bytes.
- In any line, the parameter name and value or values are separated from each other by a single space. This means that parameter names and values must not contain spaces.
- The parameter name is the first group of characters in the line.
- All parameter lines are case-sensitive.
- Free-form comments are allowed. Lines containing free-form comments are preceded by the ASCII 35 (#) symbol.

Configuration file location

smscIF.cfg is located in **/IN/service_packages/SLEE/etc**.

Parameters

smscIF.cfg contains the following parameters.

`cdrInterfaceName`

Syntax:	`cdrInterface` *`name`*
Description:	The handle of the EDR Interface process to use to write EDRs.
Type:	String
Optionality:	Optional (not used if not set).
Allowed:	Must match the handle for the EDR Interface set in **SLEE.cfg**.
Default:	
Notes:	
Example:	`cdrInterface cdrIF`

cdrLibrary

Syntax:	`cdrLibrary` *`path bin`*
Description:	The UPC library to use to generate EDRs based on SMS messages sent from smsInterface.
Type:	String
Optionality:	Optional (not used if not set).
Allowed:	
Default:	
Notes:	For more information about libsmsCdr, see *USSD GW Technical Guide*.
Example:	`cdrLibrary /IN/service_packages/UPC/lib/libsmsCdr.so`

classDefault

Syntax:	`classDefault` *`0|1|2|3`*
Description:	When smsInterface receives an SMS message over the SLEE and the message doesn't have a MessageClass set, set the class parameter to this value in the message to the SMSC.
Type:	Integer
Optionality:	Optional (default used if not set).
Allowed:	-1 Do not set a message class (leave blank). 0 1 2 3
Default:	-1
Notes:	
Example:	`classDefault 0`

connectTimeout

Syntax:	`connectTimeout` *`seconds`*
Description:	After a connection attempt, the maximum number of seconds that *smsInterface* (on page 22) will wait for a response from the SMSC before deciding that the attempt has failed.
Type:	Integer
Optionality:	Mandatory
Allowed:	
Default:	60
Notes:	
Example:	`connectTimeout 60`

failoverInterval

Syntax:	`failoverInterval` *`seconds`*
Description:	After a failed connection to the SMSC's primary IP address (*ip1* (on page 12)), the maximum number of seconds that *smsInterface* (on page 22) will wait before attempting to reconnect.
Type:	Integer

Optionality: Mandatory
Allowed:
Default: 60
Notes:
Example: `failoverInterval 60`

idleTimeout

Syntax: `idleTimeout` *`seconds`*
Description: The maximum number seconds, that *smsInterface* (on page 22) will remain inactive.
Type: Integer
Optionality: Mandatory
Allowed:
Default: 86400
Notes: 86400 seconds is one day.
Example: `idleTimeout 60`

ip1

Syntax: `ip1` *`ip port`*
Description: The primary address of the SMSC.
The port number associated with the primary address of the SMSC.
Type: *ip* is an Internet protocol number in dotted-decimal notation. For example, 125.3.57.155.
port is an integer.
Optionality: Mandatory
Allowed:
Default: None
Notes:
Example: `ip1 127.0.0.1 6666`

ip2

Syntax: `ip2` *`ip port`*
Description: The secondary address of the SMSC.
The port number associated with the secondary address of the SMSC.
Type: *ip* is an Internet protocol number in dotted-decimal notation. For example, 125.3.57.155.
port is an integer.
Optionality: Optional
Allowed:
Default: None
Notes:
Example: `ip2 127.0.0.15 6666`

loginTimeout

Syntax: `loginTimeout` *seconds*

Description: The maximum number of seconds, that *smsInterface* (on page 22) will wait for a logon message from the SMSC.

Type: Integer

Optionality: Mandatory

Allowed:

Default: 60

Notes:

Example: `loginTimeout 60`

maxSmsPerSecond

Syntax: `maxSmsPerSecond` *max*

Description: The maximum number of SMS sent to the SMS Center in one second.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 2147483647 (unlimited)

Notes: The throttling limit is calculated by measuring the number of messages sent in each 10th of a second. If the number of messages sent in the last 10 deci-seconds adds up to more than maxSmsPerSecond, we queue the message instead of sending it.

Example: `maxSmsPerSecond 2000`

msgLife

Syntax: `msgLife` *seconds*

Description: The maximum number of seconds that *smsInterface* (on page 22) will keep messages in the queue.

Type: Integer

Optionality: Mandatory

Allowed:

Default: 3600

Notes: 3600 seconds is one hour.

Example: `msgLife 3600`

password

Syntax: `password` *passwd*

Description: The password sent from *smsInterface* (on page 22) to the SMSC.

Type: String

Optionality: Mandatory

Allowed:

Default: None

Notes:

Example: `password ALPHA@NUM`

queueMaximum

Syntax:	`queueMaximum` *`max`*
Description:	The maximum number of elements that *smsInterface* (on page 22) will queue in the holding queue.
Type:	Integer
Optionality:	Mandatory
Allowed:	
Default:	5000
Notes:	For more information about message queuing, see *Message queuing* (on page 3).
Example:	`queueMaximum 5000`

retryInterval

Syntax:	`retryInterval` *`seconds`*
Description:	The maximum number of seconds after a failed connection attempt that *smsInterface* (on page 22) will wait before attempting another connection.
Type:	Integer
Optionality:	Mandatory
Allowed:	
Default:	1
Notes:	
Example:	`retryInterval 60`

retryNumber

Syntax:	`retryNumber` *`int`*
Description:	After a failed attempt to connect to connect to an IP address, the maximum number of times that *smsInterface* (on page 22) will attempt to connect to the same address again.
Type:	Integer
Optionality:	Mandatory
Allowed:	
Default:	1
Notes:	
Example:	`retryNumber 1`

tracingDirectory

Syntax:	`tracingDirectory` *`dir`*
Description:	The directory to write the tracing log to.
Type:	String
Optionality:	Optional (default used if not set).
Allowed:	
Default:	/IN/service_packages/UCP/tmp If that directory is not available, defaults to /tmp.
Notes:	New logging is appended to the end of any previous file. File is named in the following format: `ucpTrace%04u%02u%02u.log`

Example: `tracingDirectory /IN/service_packages/SMSC/tmp`

`tracingEnabled`

Syntax: `tracingEnabled` *`true|false`*
Description: Whether or not to enable tracing.
Type: Boolean
Optionality:
Allowed: true, false
Default:
Notes:
Example: `tracingEnabled false`

`userID`

Syntax: `userID` *`name`*
Description: The logon name sent from *smsInterface* (on page 22) to the SMSC.
Type: String
Optionality: Mandatory
Allowed:
Default: None
Notes:
Example: `userID ALPHA@NUM`

Configuration file example

Listed below are lines of a typical **smsclF.config** file. Comment lines have been removed.

```
loginTimeout 60
idleTimeout 60
retryInterval 60
failoverInterval 60
connectTimeout 60
ip1 127.0.0.1 6666
ip2 127.0.0.15 6666
retryNumber 1
queueMaximum 5000
msgLife 3600
userID ALPHA@NUM
password ALPHA@NUM
maxSmsPerSecond 2000
```

Note: An example **smsclF.cfg** file called **smsclF.cfg.example** is provided when SMSC is installed.

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.

notificationIF

Purpose

notificationIF is a SLEE application that sends preformed text messages from Convergent Charging Controller applications to customer's mobile telephones using smsInterface.

notificationIF receives commands in the form of SLEE events. These events determine:

- MSISDN
- Application
- Type
- Language
- A list of substitution parameters

The application, type and language are used to source the message.

The MSISDN routes the message.

The substitution parameters customize the message.

Location

This binary is located on SLCs.

Startup

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

```
INTERFACE=Notification notificationIF.sh /IN/service_packages/SLEE/bin
 UDG
```

Notes:

- notificationIF.sh is a shell script which starts the notificationIF process.
- The above are defaults and may vary.

Configuration

notificationIF reads the `notificationIF` section of the **eserv.config** file. For more general information about this file, see **eserv.config** *Configuration* (on page 9).

Parameters

Parameters of the `notificationIF` section of the **eserv.config** file are defined below.

`fromAddress`

Syntax:	`fromAddress = "`*`OriginatingAddress`*`"`
Description:	The originating address in outbound SMSs.
Type:	String
Optionality:	Optional
Allowed:	
Default:	None
Notes:	
Example:	`fromAddress = "441473289900"`

`oracleLogin`

Syntax:	`oracleLogin = "`*`UserName/Password`*`"`
Description:	The user name and password that the Notification interface must use when it connects to the Oracle database.
Type:	String
Optionality:	Mandatory
Allowed:	
Default:	/
Notes:	
Example:	`oracleLogin = "/"`

`smsIF`

Syntax:	`smsIF = "`*`handle`*`"`
Description:	The SLEE interface handle of the SMS interface.
Type:	String
Optionality:	Mandatory
Allowed:	
Default:	`smscIF`
Notes:	
Example:	`smsIF = "smsIF"`

`xmsDestNPI`

Syntax:	`xmsDestNPI = `*`indicator`*
Description:	The XMS destination numbering plan indicator.
Type:	Integer

Optionality: Optional
Allowed: Refer to the `xmsOrigNPI` parameter for permitted values for this parameter.
Default: 0
Notes:
Example: `xmsDestNPI = 0`

xmsDestTON

Syntax: `xmsDestTON = type`
Description: The XMS destination number type
Type: Integer
Optionality: Optional
Allowed: Refer to the `xmsOrigTON` parameter for permitted values for this parameter.
Default: 0
Notes:
Example: `xmsDestTON = 0`

xmsDirectFromPrefix

Syntax: `xmsDirectFromPrefix = "prefix"`
Description: The prefix that precedes the originating address when using MM Direct.
Type: String
Optionality: Optional
Allowed:
Default: None
Notes:
Example: `xmsDirectFromPrefix = "2"`

xmsFDAFromPrefix

Syntax: `xmsFDAFromPrefix = "prefix"`
Description: The prefix that precedes the originating address when using MM FDA.
Type: String
Optionality: Optional
Allowed:
Default: None
Notes:
Example: `xmsFDAFromPrefix = "1"`

xmsiWrapperIfName

Syntax: `xmsiWrapperIfName = "handle"`
Description: The SLEE interface handle of the XMSI wrapper interface.
Type: String
Optionality: Optional
Allowed:
Default: None
Notes: Messaging Manager cannot be used if this parameter is missing or left blank.

Example: `xmsiWrapperIfName = "xmsIF"`

xmsOrigNPI

Syntax: `xmsOrigNPI = indicator`

Description: The XMS originating numbering plan indicator.

Type: Integer

Optionality: Optional

Allowed: Numbering plan bits allocated to the Global Title Indicator 0011.

Bits	*type*	Significance
0000	0	Unknown
0001	1	ISDN and telephony numbering plan.
0010	2	Generic numbering plan.
0011	3	Data numbering plan.
0100	4	Telex numbering plan.
0101	5	Maritime mobile numbering plan.
0110	6	Land mobile numbering plan.
0111	7	ISDN and mobile numbering plan.
1110	14	Private network or network-specific numbering plan.

Default: 0

Notes: For information about Global Title Indicator 0011, refer to ITU-T Q.713, *Specifications of Signalling System No. 7 — Signalling Connection Control part (SCCP)*.

Example: `xmsOrigNPI = 0`

xmsOrigTON

Syntax: `xmsOrigTON = type`

Description: The XMS originating number type.

Type: Integer

Optionality: Optional

Allowed: Encoding scheme bits allocated to Global Title Indicator 0011.

Bits	*type*	Significance
0000	0	Unknown
0001	1	BCD, odd number of digits.
0010	2	BCD, even number of digits.
0011	3	National specific.

Default: 0

Notes: For information about Global Title Indicator 0011, refer to ITU-T Q.713, *Specifications of Signalling System No. 7 — Signalling Connection Control part (SCCP)*.

Example: `xmsOrigTON = 0`

xmsPC

Syntax:	xmsPC = *destination*
Description:	The XMS destination PC.
Type:	Integer
Optionality:	Optional
Allowed:	
Default:	1
Notes:	
Example:	xmsPC = 1

xmsSSN

Syntax:	xmsSSN = *destination*
Description:	The XMS destination SSN.
Type:	Integer
Optionality:	Optional
Allowed:	
Default:	1
Notes:	
Example:	xmsSSN = 1

tzType

Syntax:	tzType = *int*
Description:	Sets the timezone to be used by the Send Notification feature node when presenting dates and times to the subscriber.
Type:	Integer
Optionality:	Optional (default used if not set)
Allowed:	0 – Use system (local) timezone on the SLC 1 – Use GMT 3 – Use the timezone of the service number called 4 – Use the timezone of the logical CLI (logical calller) 5 – Use the timezone of the network CLI (calling number)
Default:	0
Notes:	The tzType = 2 setting is reserved for future use.
Example:	tzType = 1

Example notificationIF section

An example of the notificationIF section of the **eserv.config** file is listed below. Comment lines have been removed.

```
notificationIF = {

    oracleLogin = "/"
    smsIF = "smscIF"
    xmsiWrapperIfName = "xmsIf"
    fromAddress = "441473289900"
    xmsFDAFromPrefix = "1"
    xmsDirectFromPrefix = "2"
```

```
    xmsPC = 1
    xmsSSN = 1
    xmsOrigTON = 0
    xmsOrigNPI = 0
    xmsDestTON = 0
    xmsDestNPI = 0
    tzType = 0

}
```

Output

notificationIF logs errors to the file specified by *LOG_FILE* (on page 7).

smsInterface

Purpose

smsInterface provides an interface to SMS Centers to send SMS messages. It is one of two main processes in the SMSC component.

Location

This binary is located on SLCs.

Startup

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

```
INTERFACE=smscIF smscIF.sh /IN/service_packages/SLEE/bin UDG
```

Notes:

- smscIF.sh is a shell script which starts the smsInterface process.
- The above are defaults and may vary.
- In a standard configuration, only a single instance of the smsInterface will be started. Multiple instances of the smsInterface process can be run. For more information about multi-interface configuration, see *Multiple instance configuration* (on page 8).

Configuration

smsInterface is configured by the **smscIF.cfg** file. For more information about this configuration, see *Configuring the smscIF.cfg* (on page 10).

Output

smsInterface logs errors to the file specified by *LOG_FILE* (on page 7).

Chapter 4

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

SMS Interface component package

An installation of the SMS Interface component package includes the following, on the SLC:

- SMSC

Installation Prerequisites

Set SMSC database privilege

Before the installation of the SMSC a privilege in the Oracle DB must exist or, if it does not exist, must be granted.

Follow these steps to set the SMSC database privilege.

Step	Action
1	Find out which OS user(s) run the SLEE instance(s) that run the SMSC.
2	Grant execution privileges to the corresponding Oracle user(s): Enter on the command line: `su - oracle -c "ORACLE_SID=SCP sqlplus '/ as sysdba'"`

Step	Action
3	In the sqlplus session, enter on the command line: `grant execute on sys.dbms_alert to ops$ccs_oper;` **Note:** ccs_oper is the OS user.
4	Repeat steps 2 and 3 for each OS user.
5	Restart the relevant SLEE instance.
6	Repeat steps 2 through 5 for each SLEE instance.

Post-installation Configuration

Configuring the installation

When running the installation, the configuration is done automatically.

However, the configuration file for the smsInterface **smsIF.cfg** is not put in place from the package installation but is instead left with a suffix **.example**

This will not be picked up by the smsInterface.

The customer must edit this file to configure the operation for the smsInterface to their requirements.

When the file is correct it may be copied to the a new file with the name **smsclF.cfg**