Copyright

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

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

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

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

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

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

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

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

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.
## Contents

About This Document ........................................................................................................... v
Document Conventions ......................................................................................................... vi

### Chapter 1

**System Overview** ............................................................................................................. 1

- Overview .......................................................................................................................... 1
- Introduction to LCA ........................................................................................................... 1
- Message Processing .......................................................................................................... 3
- Alarms, Statistics, Reports and EDRs .............................................................................. 8

### Chapter 2

**Configuration** .................................................................................................................. 11

- Overview .......................................................................................................................... 11
- Configuration Overview .................................................................................................. 11
- eserv.config Configuration ............................................................................................. 12
- Global Configuration ....................................................................................................... 15
- SMSC Connections ......................................................................................................... 19
- Service Key Configuration ............................................................................................... 22
- sid Filter Fields ................................................................................................................. 24
- Returned Parameter Mapping .......................................................................................... 30
- LCA Control Plan ............................................................................................................ 32

### Chapter 3

**Background Processes** .................................................................................................. 33

- Overview .......................................................................................................................... 33
- ldapControlAgent .......................................................................................................... 33

### Chapter 4

**About Installation and Removal** ................................................................................... 35

- Overview .......................................................................................................................... 35
- Installation and Removal Overview ................................................................................ 35
- Checking the Installation ................................................................................................ 35
- Post-installation Configuration ....................................................................................... 37

**NCC Glossary of Terms** ................................................................................................. 39

**Index** ................................................................................................................................ 43
About This Document

Scope

The scope of this document includes all the information required to install, configure and administer the LCA application.

Audience

This guide was written primarily for system administrators and persons installing, configuring and administering the LCA application. 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 technical guide. Attempting to install, remove, configure or otherwise alter the described system without the appropriate background skills, could cause damage to the system; including temporary or permanent incorrect operation, loss of service, and may render your system beyond recovery.

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

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

Related documents

The following documents are related to this document:

- SLEX Technical Guide
- SMS Technical Guide
- ACS Technical Guide
- TCS Short Message Service Center Prepaid Messaging LDAP Interface Control Document (SMSC71LDAPINTERFACEV2.doc)
Document Conventions

Typographical Conventions

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

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

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

Introduction

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

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

In this chapter

This chapter contains the following topics.

Introduction to LCA 1
Message Processing 3
Alarms, Statistics, Reports and EDRs 8

Introduction to LCA

Introduction

The LDAP Control Agent (LCA) component provides a SLEE interface that runs on the SLC. The interface (ldapControlAgent (on page 33)) maps billing requests between LDAP and ETSI INAP. LDAP messages received from the SMSCs are converted into INAP operations. These messages are then passed over the SLEE to an application which handles INAP messages. This is usually the ACS slee_acs application.

Each LDAP request/response interaction is a real-time charging interaction controlled by the SMSC. LDAP Bind Requests are used for identification.

Functionality

The LCA is responsible for:

- Managing LDAP connections
- Accepting and verifying LDAP search requests
- Relaying valid LDAP search requests to the client SLEE application through INAP

The LDAP protocol supported by the LCA is a subset of LDAP v2.
Diagram

Here is an example of the main components of the LCA.

Profile tags

The LCA profile tags are used to temporarily store subscriber details. This table shows a list of LCA profile tag values.

<table>
<thead>
<tr>
<th>Description</th>
<th>Decimal</th>
<th>Type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCA Units</td>
<td>7077889</td>
<td>Integer</td>
<td>The number of messages for which the subscriber’s account is to be debited. Typically, this will be the single digit “1”. Usually relevant for debit messages only. Populated if the tag (on page 30) parameter points to this profile tag number.</td>
</tr>
<tr>
<td>LCA Amount</td>
<td>7077890</td>
<td>Integer</td>
<td>An amount of money (to be credited to the subscriber’s balance). Specified in natural terms (for example: 14.20). Usually relevant for credit messages only, and usually needs to be associated with a scale of 100. Populated if the tag (on page 30) parameter points to this profile tag number.</td>
</tr>
<tr>
<td>LCA Currency Label</td>
<td>7077891</td>
<td>String</td>
<td>The currency label indicating the currency the SMSC is working in. Usually a string such as</td>
</tr>
</tbody>
</table>
Description | Decimal | Type | Remarks
--- | --- | --- | ---
“USD”. Populated if the tag (on page 30) parameter points to this profile tag number.
LCA Subscriber Balance | 7077892 | Integer | Stores a subscriber’s balance after a billing action. Specified by balanceProfileTag (on page 17).

For more information about profile tags, see *CPE User’s Guide*.

**TCS LDAP**

The TCS-specific LDAP-based protocol is a subset of the LDAP v2 protocol defined by RFC 1777. It is defined in *TCS Short Message Service Center Prepaid Messaging LDAP Interface Control Document* (SMSC71LDAPINTERFACEV2.doc).

The TCS implementation defines constraints which limit or specialize the scope of the implementation but do not contradict the LDAP specifications (except in the one case described below). TCS LDAP defines the following limitations and extensions for this interface:

- Only use BindRequest/Response, SearchRequest/ResponseEntry/ResponseResult
- No authentication except simple name/password
- No nested filters are permitted
- Only a single filter attribute ("sid")
- A proprietary “colon-separated” string for the "sid" filter attribute value
- Specific hard-coded returned attributes.

There is one non-compliant LDAP implementation feature defined in TCSE LDAP. The standard LDAP v2 says that every searchResponseEntry MUST be followed by a searchResponseResult. However the TCS LDAP says that searchResponseEntry is NEVER followed by a matching searchResponseResult. So, when the LCA LDAP server operates as required by TCS, it is not compatible with any standards-based LDAP client APIs. For this reason, the LCA is capable of operating in TCS-compliant or Strict-LDAP mode. The mode to use is defined by the strictLdapMode (on page 18) parameter.

**Message Processing**

**Messages**

This table describes the LDAP messages supported by the LCA.

<table>
<thead>
<tr>
<th>Name</th>
<th>Message Flow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BindRequest</td>
<td>SMSC to LCA</td>
<td>The BindRequest message is sent to authorize an SMSC to send messages to ldapControlAgent (on page 33). For more information, see <em>Establishing a SMSC connection</em> (on page 4).</td>
</tr>
<tr>
<td>BindResponse</td>
<td>LCA to SMSC</td>
<td>The BindResponse message provides a response code to the SMSC. For more information, see <em>Establishing a SMSC connection</em> (on page 4).</td>
</tr>
<tr>
<td>SearchRequest</td>
<td>SMSC to LCA</td>
<td>The SearchRequest message is used by the SMSC send a request to LCA.</td>
</tr>
<tr>
<td>Name</td>
<td>Message Flow</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information, see <code>SearchRequests</code> (on page 7).</td>
</tr>
<tr>
<td>SearchResponseEntry</td>
<td>LCA to SMSC</td>
<td>SearchResponseEntry message is returned after a SearchRequest. This should cover successful and unsuccessful execution on slee_acs.  For more information, see <code>ServiceResponseEntries</code> (on page 7).</td>
</tr>
<tr>
<td>SearchResponseResult</td>
<td>LCA to SMSC</td>
<td>For LCA error cases, a SearchResponseResult message is returned to the SMSC with an appropriate non-zero “result” code. These errors are related to message decode failures or internal LCA system failures. For more information, see <code>ServiceResponseResults</code> (on page 8).</td>
</tr>
</tbody>
</table>

**LDAP-INAP message mapping**

This table shows the mapping between LDAP SearchRequest/SearchResponse and INAP messages.

<table>
<thead>
<tr>
<th>LDAP message</th>
<th>INAP message</th>
</tr>
</thead>
<tbody>
<tr>
<td>SearchRequest</td>
<td>InitialDP</td>
</tr>
<tr>
<td>SearchResponseEntry (res == “00”)</td>
<td>Connect</td>
</tr>
<tr>
<td>SearchResponseEntry (res &lt;&gt; “00”)</td>
<td>ReleaseCall</td>
</tr>
<tr>
<td>SearchResponseResult (result != 0)</td>
<td>ABORT/Continue/Other/Internal Error</td>
</tr>
</tbody>
</table>

**Note:** When slee_acs wishes ldapControlAgent to reply with SearchResponseEntry with “res” != “00” it should use ReleaseCall. The ReleaseCall cause is mapped to the “res” attribute value via the configuration option `releaseMap` (on page 31).

**Establishing a SMSC connection**

When an SMSC connects to LCA it uses a BindRequest to establish its identity to lcapControlAgent. The name provided in the BindRequest identifies the SMSC, and includes:
- The “logical server ID” of the SMSC
- The “service type” to authorise (for more information about service types, see `Search request types` (see “SearchRequests” on page 7))
- A password (this is then compared by the lcapControlAgent with its own password list)

ldapControlAgent responds with a BindResponse to finalise the connection. If the response code is 0, then the SMSC is successfully authenticated. Typically the SMSC/ldapControlAgent socket will remain connected, but ldapControlAgent will fail any further bind attempt on the connection. Once this message pair have been completed, the SMSC can begin to perform billing interactions through lcapControlAgent to slee_acs and finally through to the subscriber’s specific Voucher and Wallet Server.

**Note:** Each service connection is associated with a single permitted type of request (debit, credit or balance). More than one service connection can run over a single TCP (socket) connection. For more information about configuring connections, see `SMSC Connections` (on page 19).
Message flow

LCA will translate all valid LDAP search requests before sending them to slee_acs. The following table describes an example message flow:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | The SMSC sends an LDAP SearchRequest to *ldapControlAgent* (on page 33) on a SLC over an established connection.  
   **Note:** The SearchRequest type will match the connection type.  
   For more information about how connections are established, see *Establishing a SMSC connection* (on page 4). |
| 2    | *lcapControlAgent* extracts the parameters (for example, called party, calling party, number of units/messages) from the colon-separate "sid" filter attribute in the SearchRequest.  
   *lcapControlAgent* populates a new InitialDP from the parameters as specified by the idpMapping configuration section. Some parameters are placed in the standard InitialDP fields. Other parameters (for example, number of units/messages) are placed in extension fields in the special ACS "Incoming Profile" extension block. When the system can successfully convert a SMSC request into an InitialDP, the mapping of data is based on the configuration.  
   For more information about idpMapping, see *sid Filter Fields* (on page 24). |
| 3    | *ldapControlAgent* sends the InitialDP to slee_acs over the SLEE.  
   SLEE.cfg configuration determines which SLEE application to send the InitialDP to. It is usually slee_acs.  
   If the service uses slee_acs, it will load a dedicated ACS control plan. The control plan will generally use a CCS billing node to query a Voucher and Wallet Server for subscriber details. |
| 4    | slee_acs extracts the profile block and makes it available to the control plan.  
   Nodes can now access all parameters that have been mapped to an IDP parameter.  
   Any nodes that read profile block information can retrieve profile information from the Call Context profile block, and the Incoming Extensions profile block. For more information about profile blocks, see *Profile tags* (on page 2). |
| 5    | slee_acs sends a billing request to a VWS over an interface. |
| 6    | The Voucher and Wallet Server replies indicating success or failure request. The success case must include the subscriber's balance. |
| 7    | slee_acs (or other SLEE application if configured) returns an INAP ReleaseCall or Connect operation to ldapControlAgent.  
   If the request was a debit, slee_acs writes the new balance into a profile field in the pre-agreed ACS "Outgoing Profile" extension block in the Connect message. If slee_acs uses a Connect, it should always intend that SearchResponseEntry with "res" = "00" is returned. slee_acs should always provide a balance value in the Connect extension. |
| 8    | *ldapControlAgent* translates this INAP response into the appropriate LDAP response for the SMSC.  
   The *ldapControlAgent*:  
   - Receives the Connect  
   - Scales and reformats it according to the *balanceScale* (on page 17) and *balancePrecision* (on page 18) parameters  
   - Reads the balance from the Connect Extension Profile, using the profile field with the tag configured by the *balanceProfileTag* (on page 17) configuration |
Step | Action
--- | ---
option
- Constructs a SearchResponseEntry which includes:
  - An "objectName" (a copy of the full "sid" filter attribute string copied from the request message "objectName")
  - At least one of these attribute fields: "res", "bal", "bal1", "bt1", and/or "cur"

For more information about SearchResponseEntries, see *ServiceResponseEntries* (on page 7).

**Successful message**

This message flow diagram shows a successful message flow.

**Soft error message flow**

This message flow diagram shows a soft error.
Hard error message flow

This message flow diagram shows a hard error.

![Hard error message flow diagram](image)

SearchRequests

Which request the SearchRequest message is for depends on the message's "BaseObject" field, which includes the "logical server ID" as well as the "service type". The "BaseObject" must be identical to the "name" used in the initial BindRequest. lcapControlAgent enforces this.

This table describes the three supported SearchRequest types.

<table>
<thead>
<tr>
<th>SearchRequest</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debit request</td>
<td>Request a debit action against a subscriber account. Used for one or more SMS messages.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Reservations are not supported.</td>
</tr>
<tr>
<td>Credit request</td>
<td>Request a credit action for a subscriber account. Used for a specified amount of currency. This includes refund for undelivered message.</td>
</tr>
<tr>
<td>Balance request</td>
<td>Used to determine a subscriber account's current primary balance value.</td>
</tr>
</tbody>
</table>

A single "sid" attribute in the "filter" tree of the SearchRequest includes all the information necessary for the search request (such as the called and calling party numbers).

lcapControlAgent retrieves all the necessary data from this "filter" field. For more information, see sid Filter Fields (on page 24). This "sid" filter field is a colon-separate string, which is very different from a conventional LDAP request string. Many key pieces of information are available in normal IDP fields such as the calling and called party number fields. Hence many standard nodes (for example: the CCS Named Event node) will work directly on this data. The mapping from colon-separated field to profile field is different for debit, credit and balance.

ServiceResponseEntries

ServiceResponseEntry messages are triggered by messages returned by slee_acs.

When the SearchRequest is handled successfully by the client application, the "res" field is set to "00" and, one or more of the other attribute fields are included: "res", "bal", "bal1", "bt1", and/or "cur". The fields included depends on the SearchRequest type.
When the system does not successfully handle a SearchRequest, a "soft error" occurs. A soft error is a SearchResponseEntry with "res" != "00". Soft errors, do not have any other attributes. Soft errors are varied and include all subscriber-based errors. Soft errors can be caused by:

- Subscriber Unknown
- Insufficient Funds
- Invalid Subscriber Number
- Subscriber Account Frozen

Calls which are sent to slee_acs should only result in a ServiceResponseEntry showing either success, or a "soft error" at most.

**ServiceResponseResults**

The SearchResponseResult is used only for hard errors (SMSC authentication errors, LCA internal system failure, SLEE errors). If the system is well-configured, then a call which triggers an InitialDP to slee_acs should never result in a SearchResponseResult (hard error) being sent to the SMSC.

**Alarms, Statistics, Reports and EDRs**

**Alarms**

LCA uses the SMS integrated alarms collection, viewing and forwarding system. The alarms generated by all components of LCA are consolidated on the SMS and stored in a centralized alarm database.

The operator can:

- View the alarms through the alarm viewer built into the SMS screens
- Forward all alarms to an integrated external fault management system using SNMP v1 or v3

Alarms can be automatically deleted from the SMF alarm database after a configurable period.

For more information about the:

- Specific alarms generated by LCA, see *LCA Alarms Guide*
- SMS alarms subsystem, see *SMS User's Guide*

**Statistics**

This table shows the statistics recorded by LCA. Statistics are defined around:

- The two types of messages that can be received (BindRequest and SearchRequest), and
- (For SearchRequest messages) the three types of services supported (debit, credit and balance).

**Note:** All statistics are logged by *ldapControlAgent* (on page 33) during normal operation and are available for SMS replication via the replication group "SMF_STDEF_LCA". The group is available under the SMS application after the lcaSms package is installed.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUM_BIND_REQUESTS</td>
<td>The total number of BindRequest messages received that have been able to be parsed by ldapControlAgent. The &quot;detail&quot; of the statistic will be set to the logical server ID value that is read from the BindRequest message. If the logical server ID could not be read, then the &quot;detail&quot; of the statistic will be set to &quot;unknown&quot;.</td>
</tr>
<tr>
<td>NUM_BIND_RESPONSES</td>
<td>The total number of successful BindResponse messages sent. A BindResponse message is successful when the SMSC has been successfully authenticated and the response includes a</td>
</tr>
<tr>
<td>Statistic</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>response code of 00. The “detail” of the statistic will be set to the value used for equivalent the TOTAL BIND REQUEST statistic.</td>
<td></td>
</tr>
</tbody>
</table>
| NUM_BIND_ERROR_RESPONSES | The number of BindResponse sent by the system where the result code is non-zero. This statistic is incremented when:  
  - Authentication of a server fails  
  - The server ID and service could not be read from the bind request's 'name' element  
NUM_BIND_REQUESTS = NUM_BIND_RESPONSES + NUM_BIND_ERROR_RESPONSES. |
| NUM_DEBIT_SERVICE_REQUESTS | The total number of debit service request messages received by ldapControlAgent. The “detail” of the statistic will be set to the logical server ID read from the request message.  
Both MO and MT service requests increment this statistic. |
| NUM_DEBIT_SERVICE_RESPONSES | The total number of debit service response messages sent by ldapControlAgent. The “detail” of the statistic will be set to the logical server ID of the corresponding request message.  
**Note:** This statistic is only incremented by responses which are generated from a successful INAP ReleaseCall or Connect message. |
| NUM CREDIT SERVICE REQUESTS | The total number of credit service request messages received by ldapControlAgent. The “detail” of the statistic will be set to the logical server ID. |
| NUM CREDIT SERVICE RESPONSES | The total number of credit service response messages sent by ldapControlAgent. The “detail” of the statistic will be set to the logical server ID of the corresponding request message.  
**Note:** This statistic is only incremented by responses which are generated from a successful INAP ReleaseCall or Connect message. |
| NUM_REQUEST_BALANCE_REQUESTS | The total number of request balance request messages received by ldapControlAgent. The "detail" of the statistic will be set to the logical server ID. |
| NUM_REQUEST_BALANCE_RESPONSES | The total number of request balance response messages sent by ldapControlAgent. The "detail" of the statistic will be set to the logical server ID of the corresponding request message.  
**Note:** This statistic is only incremented by responses which are generated from a successful INAP ReleaseCall or Connect message. |
| NUM UNKNOWN REQUESTS | If ldapControlAgent is unable to parse or identify a request that comes in (that is, it is not a bind or a search request) this statistic is incremented.  
Neither the NUM BIND_ERROR RESPONSES nor the NUM_SEARCH_ERROR RESPONSES are incremented as no response is sent back in such cases. |
| NUM UNKNOWN SEARCH REQUESTS | If a search request comes in on an unauthenticated connection or the type of request for a search request cannot be identified, then this statistic is also incremented. |
Chapter 1

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the logical service ID can or has be identified, then it is stored in the detail field, otherwise the statistic is incremented with a detail field of 'unknown'.</td>
</tr>
<tr>
<td>NUM_SEARCH_ERRORRESPONSES</td>
<td>The number of search error responses sent by the system. An error response is sent for configuration errors, when timeouts occur waiting for responses to IDPs sent, or when a search request cannot be successfully converted into an IDP.</td>
</tr>
</tbody>
</table>

**Reports**

LCA does not install any specific SMS reports. However, you can report on any statistics which are recorded using the SMS Application report on the Service Management System Report Functions screen. For more information about running SMS reports, see *SMS User’s Guide*.

LCA provides a status report. For more information about how to run this report, see *Status reports* (on page 34).

**EDRs**

LCA does not write any EDRs itself. However messages sent from ldapControlAgent to slee_acs will result in an EDR being written by slee_acs. For more information about what EDRs are written by slee_acs, see *ACS Technical Guide*. 
Chapter 2

Configuration

Overview

Introduction

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

In this chapter

This chapter contains the following topics.

Configuration Overview

Introduction

This topic provides a high level overview of how the LCA component is configured.

Configuration components

LCA is configured by the following components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Locations</th>
<th>Description</th>
<th>Further Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>eserv.config</td>
<td>All SLC machines</td>
<td>The LCA is configured in the ‘LCA’ section of the eserv.config file.</td>
<td>eserv.config Configuration (on page 12)</td>
</tr>
<tr>
<td>SLEE.cfg</td>
<td>All SLC machines</td>
<td>The SLEE configuration must be altered to include the LCA.</td>
<td>SLEE Technical Guide</td>
</tr>
<tr>
<td>acs.conf</td>
<td>All SLC machines</td>
<td>The ACS framework must be configured to accept LCA calls.</td>
<td>ACS Technical Guide</td>
</tr>
<tr>
<td>SMF database</td>
<td>SMS</td>
<td>Statistics, profile block and EFM alarms configuration for LCA. Configured automatically when lcaSms is installed.</td>
<td>Installing lcaSms on a SMS</td>
</tr>
<tr>
<td>ACS control plans</td>
<td>SMS</td>
<td>Specific control plans must be developed to handle LCA requests successfully.</td>
<td>LCA Control Plan (on page 32)</td>
</tr>
</tbody>
</table>
eserv.config Configuration

Introduction

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

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

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

Configuration file format

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

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

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

```
{ 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, ldapControlAgent (on page 33) (which runs LCA on the SLC) 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: \[ \text{ESERV_CONFIG_FILE} = "path/file" \]

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

Type: String

Optionality: Optional (default used if not set).

Allowed: 

Default: /IN/service_packages/eserv.config

Notes:

Example:

Editing the file

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

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

Loading eserv.config configuration changes

To cause ldapControlAgent to reread its configuration from the eserv.config configuration file, send a SIGHUP signal to the ldapControlAgent process. This will reload all configuration, including all connection configuration. Existing connections will not be affected. This allows you to make configuration changes and have them take effect without affecting any existing connections. If the configuration reload fails due to a problem with the configuration file, the current working configuration will continue to be used.

Notes:

* Connected servers will not be disconnected when the configuration is reloaded. The ports and network addresses the agent listens on will be altered to match the new configuration so that future connections must meet the new configuration.
* If a password has been changed or removed, any existing LDAP connections will be allowed to continue. Only new connections will be affected.

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

This file shows an example of the LCA section of the eserv.config file.

\begin{verbatim}
LCA {
    listen = [
        { ip="10.42.2.50", port=1500 }
        { ip="10.42.2.50", port=1600 }
        { ip="10.42.2.50", port=1700 }
    ]
}\end{verbatim}
authentication = [
    { serverID="server1", service="debit", password="abcd1234" }
    { serverID="server1", service="credit", password="abcd1234" }
    { serverID="server1", service="balance", password="abcd1234" }
]

dialogTimeout = 0
defaultServiceKey = 108001
idpMapping = {
    debitMO = {
        #serviceKey = 108002
        #serviceKeys = [
        #    { serverID = "smsc1", serviceKey = 108003 }
        #    { serverID = "smsc2", serviceKey = 108004 }
        #]
        calledPartyNumberField = 11
        callingPartyNumberField = 3
        locationNumberField = -1
        originalCalledPartyIDField = -1
        eventTypeBCSM = 3
        redirectingPartyIDField = -1
        additionalCallingPartyNumberField = 0
        extensionTags = {
            { source = 8, type = "integer", scale = 1, tag = 7077889 }
        }
    }
    debitMT = {
        calledPartyNumberField = 3
        callingPartyNumberField = 11
        eventTypeBCSM = 12
        additionalCallingPartyNumberField = 0
        extensionTags = {
            { source = 8, type = "integer", scale = 1, tag = 7077889 }
        }
    }
    credit = {
        callingPartyNumberField = 3
        eventTypeBCSM = 3
        additionalCallingPartyNumberField = 0
        extensionTags = {
            { source = 6, type = "integer", scale = 100, tag = 7077890 }
            { source = 7, type = "string", tag = 7077891 }
        }
    }
    balance = {
        callingPartyNumberField = 3
        eventTypeBCSM = 3
        additionalCallingPartyNumberField = 0
        extensionTags = {
            { source = 6, type = "string", tag = 7077891 }
        }
    }
}
currencyCode = "USD"
balanceCode = "0"
balanceProfileTag = 7077892
balanceScale = 0.01
balancePrecision = 4
# shuffleProfileIntegers = false
# strictLdapMode = false
releaseMap = [
{  cause = 31, result = 01  
{  cause = 10, result = 02  
{  cause = 11, result = 04 
{  cause = 12, result = 05  
{  cause = 13, result = 06 

}  

defaultReleaseCallResult = 99

}

Global Configuration

Introduction

This section describes the global configuration used by ldapControlAgent.

Configuration

This text shows the the LCA global parameters supported in eserv.config.

LCA = {
  defaultServiceKey = key
  dialogTimeout = secs
  currencyCode = "str"
  balanceCode = "code"
  balanceProfileTag = tag
  balanceScale = int
  balancePrecision = int
  shuffleProfileIntegers = true|false
  strictLdapMode = true|false
}

For an example of a full LCA section of eserv.config file, see Example eserv.config file (on page 13).

Parameters

Here are the global parameters supported in the LCA section of the eserv.config file.

defaultServiceKey

Syntax: defaultServiceKey = key

Description: The default SLEE service key to which IDP messages are sent.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: This service key must match a service key defined in the SLEE.cfg file, using a configuration line such as:

SERVICEKEY=INTEGER 108001 LCA_General

For more information about SLEE service keys and SLEE.cfg, see SLEE Technical Guide.

Default: 100

Notes: Each INAP message must have a service key which matches a service key in SLEE.cfg. For more information about the ways service keys can be set, see Service Key Configuration (on page 22).

The service key configuration can be overridden for a particular message type by the serviceKey (on page 23) and serviceKeys (on page 23) parameters.

Example: defaultServiceKey = 801001
dialogTimeout

Syntax:  
dialogTimeout = seconds

Description:  The approximate number of seconds ldapControlAgent will wait for a response after sending an IDP to slee_acs. If this number of seconds is exceeded, ldapControlAgent will:

- Send an LDAP failure response back to the SMSC which sent the initial LDAP message
- Discard any response from slee_acs for the request

Type:  Integer

Optionality:  Optional (default used if not set).

Allowed:  

positive integer  The number of seconds before timing out.

0  No timeout checking will be used by ldapControlAgent. Instead, other parts of the system is expected to perform time-out checking using other methods (such as slee_acs and the Voucher and Wallet Server interface).

Default:  0

Notes:  The system uses a 10-second block based algorithm for managing timeouts. Time is split into 10 second blocks, and the timeout is placed in the next 10-second block that is at least dialogTimeout seconds in the future.

Example:  Assume the dialogTimeout is set to 5 seconds.

- 10:00:00 - start of timeout block 1
- 10:00:10 - start of timeout block 2
- 10:00:20 - start of timeout block 3
- 10:00:30 - start of timeout block 4
- 10:00:40 - start of timeout block 5
- 10:00:50 - start of timeout block 6

If a message is sent to slee_acs at 10:00:11, then the minimum dialog time out would be 10:00:16. The next timeout block though is at 10:00:20, so the timeout will be at 10:00:20.

If a message is sent to slee_acs at 10:00:16, then the minimum dialog time out would be 10:00:21. However the next timeout block is at 10:00:30, so the dialog will time out at 10:00:30, not at 10:00:21.

Example:  
dialogTimeout = 5

currencyCode

Syntax:  
currencyCode = "code"

Description:  The three-letter currency code to use globally in SearchResponse messages as the value for the cur attribute.

Type:  String

Optionality:  Optional (default used if not set).

Allowed:  A short string

Default:  "USD"

Notes:  For more information about SearchResponse messages, see Message Processing (on page 3).

Example:  currencyCode = "AUD"
balanceCode
Syntax: balanceCode = "code"
Description: The value to globally set the search result message attribute bt1 (primary balance code) to.
Type: String
Optionality: Optional (default used if not set).
Allowed: Single character string
Default: 0
Notes: It is recommended to use the default value, as the protocol expects bt1 to be the single character string "0".
Example: balanceCode = "0"

balanceProfileTag
Syntax: balanceProfileTag = tag
Description: The profile tag to read the subscriber's balance from when translating a message from slee_acs to the SMSC after a successful debit, credit or balance enquiry.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 7077892
Notes: This tag number corresponds to the LCA Subscriber Balance profile field installed by lcaSms. This tag will have been populated by slee_acs when a subscriber enquiry response was received from the Voucher and Wallet Server. This tag is set as a decimal integer. Profile tags are sometimes set in hexadecimal. For more information about LCA profile tags, see Profile tags (on page 2).
Example: balanceProfileTag = 7077892

balanceScale
Syntax: balanceScale = num
Description: Indicates the number by which the balance received from slee_acs is multiplied before placing the balance in response messages to the SMSC.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 0.01
Notes: This is used during the INAP to LDAP translation to convert the integers of balance amounts provided by slee_acs into larger strings expected by the SMSC. This parameter is used in balance scaling and rounding.
Example: slee_acs provides balances as an integer in “small money”, such as 1420 (cents). The SMSC expect the result as a string in “big money” such as “14.2000”.
Example: balanceScale = 0.01
balancePrecision

Syntax: balancePrecision = int

Description: The precision used when converting the integers that make up the balance value (big money) into a string.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: 4

Default: 4

Notes: The TCS standard says that four decimal places should always be included. This parameter is used in balance scaling and rounding.

Example: balancePrecision = 4

shuffleProfileIntegers

Syntax: shuffleProfileIntegers = true|false

Description: Determines if the integer values must be shuffled when reading/ writing integers into profile blocks sent to or retrieved from slee_acs.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed: true, false

Default: false

Notes: This will generally need to be left at the default. This is applicable only when running ldapControlAgent on x86 hardware. It is not required on SPARC hardware.

Example: shuffleProfileIntegers = false

strictLdapMode

Syntax: strictLdapMode = true|false

Description: Determines whether to enable strict LDAP v2 mode for compatibility with real-world LDAP v2 clients.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed: true Strict LDAP v2 compatibility mode. ldapControlAgent always sends a SearchResultResponse (result code 0) to follow a SearchResultEntry. Use for testing with third-party LDAP client software.
false TCS-compliant mode. Use this setting for TCSE. ldapControlAgent will never send a SearchResultResponse (result code 0) after a SearchResultEntry.

Default: false

Notes: This may need to be true on some test systems. Generally, production system should have this set to false.

Example: strictLdapMode = false
defaultReleaseCauseResult

Syntax: \[defaultReleaseCauseResult = val\]

Description: If ldapControlAgent receives a ReleaseCall INAP message from slee_acs with a cause value not mentioned in the releaseMap (on page 31) configuration the value given by this configuration option is used instead.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: Positive integer

Default: 99

Notes: Overridden by releaseMap configuration.

Example: defaultReleaseCauseResult = 32

SMSC Connections

Introduction

The configuration of the SMSC connection information is split into two parts:

1. **listen** (on page 20) configuration, the TCP/IP ports and IP addresses the ldapControlAgent (on page 33) listens on for incoming connections.
2. **authentication** (on page 21) configuration, the information used to authenticate incoming SMSC connections (servers).

Note: There is no correlation between IP addresses and ports being listened on, and valid authentication configuration. A configured server (as defined in the authentication section) can connect on any address/port combination. More than one connection can operate on a single port.

For more information about how connections are established, see *Establishing a SMSC connection* (on page 4).

Configuration

Here is an example of the LCA section which configures SMSC connections.

```
LCA = {
    listen = [
        { ip="ip", port=port }
    ]

    authentication = [
        { serverID="id", service="str", password="pwd" }
    ]
}
```

Parameters

Here are the SMSC connection parameters supported by the LCA section of eserv.config.
**listen**

**Syntax:**

```
listen = [
    { ip="ip", port=port }
    ...
]
```

**Description:**
Contains the parameters that determine the IP addresses and ports which `ldapControlAgent` (on page 33) will listen for incoming connections on.

**Type:** Array

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

**Allowed:**
If not specified, ldapControlAgent will listen on all network interfaces on the machine.

**Notes:**
There is no limit to the number of combinations which can be configured, however none of the combinations can overlap.

See also definitions for the `ip` (on page 20) and `port` (on page 20) parameters.

**Example:**

```
listen = [
    { ip="10.42.2.50", port=1500 }
]
```

**ip**

**Syntax:**

```
ip="ip"
```

**Description:**
The IP address `ldapControlAgent` (on page 33) listens for incoming LDAP requests on.

**Type:** String

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

**Allowed:** IPv4 address

**Default:** 0.0.0.0

**Notes:**
To see an example of this parameter being used in context, see `listen` (on page 20).

**Example:**

```
To see an example of this parameter being used in context, see `listen` (on page 20).
```

**port**

**Syntax:**

```
port=port
```

**Description:**
The machine port number `ldapControlAgent` (on page 33) listens on for incoming LDAP requests.

**Type:** Integer

**Optionality:** Mandatory

**Allowed:**
No default.

**Notes:**
Port numbers below 1000 are not generally usable since the SLEE (and therefore ldapControlAgent) does not run as root.

**Example:**

```
To see an example of this parameter being used in context, see `listen` (on page 20).
```

20 NCC LDAP Control Agent Technical Guide
authentication

Syntax:    

    authentication = [  
        { serverID="id"  
          service="str"  
          password="pwd"  }  
    ...  
    ]  

Description: Contains the parameters that authenticate servers that can connect to
ldapControlAgent (on page 33).

Type: Array

Optionality: Mandatory

Allowed: Add as many authentication name/password pairs as required. This will only be
limited by available memory.

Default:

Notes: Each server that can connect to the ldapControlAgent must be defined in this
section. Each server listed must include all three parameters (server, service and
password).

If a server can connect with all three services, then the server must be listed three
times, once for each service.

This is the authentication section used by the BindRequest LDAP msg.

See also definitions for the serverID (on page 21), service (on page 21) and
password (on page 22) parameters.

Example:

    authentication = [  
        { serverID="server1", service="debit", password="abcd1234"  }  
    ]

serverID

Syntax:  

    serverID = "id"

Description: The logical ID of the server provided in the incoming BindRequest message.

Type: String

Optionality: Mandatory

Allowed: String of 18 characters or less.

Default: No default

Notes: If a server can connect with all three services, then the server must be listed three
times, once for each service.

Example: To see an example of this parameter being used in context, see authentication (on
page 21).

service

Syntax:  

    service = "service"

Description: The type of service the BindRequest message is sent for.

Type: String

Optionality: Mandatory

Allowed: Allowed values are:

    - "debit"
    - "credit"
    - "balance"

Default: No default
Notes:
Example: To see an example of this parameter being used in context, see authentication (on page 21).

password
Syntax: password = "password"
Description: The password passed to LdapControlAgent (on page 33) in the BindRequest message. This is then compared by LdapControlAgent with its own password list.
Type: String
Optionality: Mandatory
Allowed: No default
Default: No default
Notes: To see an example of this parameter being used in context, see authentication (on page 21).

Service Key Configuration

Introduction
When LdapControlAgent sends a request to slee_acs it must specify a SLEE service key.

Service key parameter relationships
LCA supports three levels of specificity. LdapControlAgent checks the most specific configuration for a match first, the level two, and then to the default.

1. The most specific configuration is a serverID and message type match, configured in LCA.serviceType.serviceKeys (on page 23) parameter.
2. Service keys can also be set at the service type level (debitMO, debitMT, credit, balance query) using the LCA.serviceType.serviceKey (on page 23) parameter.
3. If no service key is configured at the message type level, LdapControlAgent will use the global service key defined by the LCA.defaultServiceKey (on page 15) configuration parameter.

Warning: If no service key configuration is set, the default for defaultServiceKey will be used, and all messages will have a service key of 100. Unless the SLEE.cfg and acs.conf files also use this service key for all LCA traffic, this will result in lost calls.

Note: The "service" for SMSC authentication purposes is "debit", "credit", "balance". The service type for service key configuration is more specific and supports "debitMO", "debitMT", "credit", "balance".

Service key correlation
The serviceKey in the INAP message is correlated between the three locations described in this table.

<table>
<thead>
<tr>
<th>Location</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>The LCA configuration in eserv.config.</td>
<td>Determines which service key to use in LDAP to INAP translation.</td>
</tr>
<tr>
<td>The SLEE configuration in SLEE.cfg.</td>
<td>Determines what SLEE service handle is used, and which application (typically slee_acs) will receive these calls.</td>
</tr>
<tr>
<td>The slee_acs configuration in acs.conf.</td>
<td>Determines how slee_acs handles requests with the corresponding SLEE service handle mapped in SLEE.cfg.</td>
</tr>
</tbody>
</table>
Service key parameters

IdapControlAgent supports the following parameters in the idpMapping subsection of eserv.config.

Note: IdapControlAgent also supports the defaultServiceKey (on page 15) parameter.

**serviceKey**

**Syntax:**

\[
\text{serviceKey} = \text{skey}
\]

**Description:**
The value to set the service key to in the outgoing message to all SMSCs for the service type this parameter is in.

**Type:**

Integer

**Optionality:**

Mandatory

**Allowed:**

**Default:**

defaultServiceKey (on page 15)

**Notes:**

Overridden by serviceKeys (on page 23) configuration.

To set a different service key for different SMSC connections, serviceKeys.

The service type can be one of:

- debitMO
- debitMT
- credit
- balance

**Example:**

**serviceKeys**

**Syntax:**

\[
\text{serviceKeys} \left[ \begin{array}{c}
\{ \text{serverID} = \text{id}, \text{serviceKey} = \text{skey} \} \\
\ldots
\end{array} \right]
\]

**Description:**
The value to set the service key to in the outgoing message to all SMSCs, depending on which SMSC connection is being used.

A list of servers, each must have these two elements:

- The serverID is the "logical server ID" and should match a server ID defined in the "authentication" section (though the LCA does not enforce this).
- The serviceKey is the integer service key to use.

**Type:**

Array

**Optionality:**

Optional (default used if not set).

**Allowed:**

**Default:**

serviceKey (on page 23)

**Notes:**

Overrides serviceKey configuration.

serverID must be 18 characters or less.

If only one service key should in outgoing messages from this type of IDP, use serviceKey.

**Example:**
sid Filter Fields

Introduction

All LDAP SearchRequest messages from the SMSC contain a filter field. This filter field contains a single attribute named "sid". The value of this "sid" filter attribute contains the event parameters which slee_acs requires in order to perform the appropriate billing interaction. The task for ldapControlAgent is to transfer these parameters into an InitialDP for slee_acs. This is specified in the idpMapping section of eserv.config.

sid string format

The actual number, position, and interpretation of each parameter depends on the specific service being executed. The sid string contains several different pieces of information, separated by a colon ":" character. To understand the mapping, refer to the TCSE specification [Document 3], and to associated system specification documents such as the nTelos SRS.

Example: This text shows a "sid" filter field for a debit MO message.

```
277339716::88:0064211617993:::0064211617993
```

Here is the TCSE definition of the fields for the debit MO scenario.

```
subscriberID::transactionID:requestingSystem:merchantID::service:content:units:balanceIndicator:debitType:destinationAddress:::internationalIndicator:connectedDNlist
```

Notes:
- In the LDAP message sent to the SMSC, the string is prefixed by "sid=".
- Some fields are empty, others are optional. Some have specific hard-coded values.

sid fields

This table shows request fields by message type. It is derived from [Document 3]. The "Index" column indicates the position of the field within the colon-delimited "filter" string of the incoming message.

<table>
<thead>
<tr>
<th>Index</th>
<th>Debit (MO + MT)</th>
<th>Credit</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>subscriberID</td>
<td>subscriberID</td>
<td>subscriberID</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>transactionID</td>
<td>transactionID</td>
<td>transactionID</td>
</tr>
<tr>
<td>3</td>
<td>requestingSystem</td>
<td>requestingSystem</td>
<td>requestingSystem</td>
</tr>
<tr>
<td>4</td>
<td>merchantID</td>
<td>merchantID</td>
<td>merchantID</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>service</td>
<td>transactionAmount</td>
<td>currencyLabel</td>
</tr>
<tr>
<td>7</td>
<td>content</td>
<td>currencyLabel</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>units</td>
<td>balanceIndicator</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>balanceIndicator</td>
<td>creditType</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>debitType</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Request fields

This table describes the function of each field.

**Note:** All fields are expected to use ASCII characters or hex digits.

<table>
<thead>
<tr>
<th>Field</th>
<th>Used by</th>
<th>Format</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>subscriberID</td>
<td>All Types</td>
<td>Up to 16 hex digits</td>
<td>The MDN (mobile directory number).</td>
</tr>
<tr>
<td>transactionID</td>
<td>All Types</td>
<td>Up to 10 characters</td>
<td>A &quot;mostly-unique&quot; transaction ID given by the SMSC. This is not used specifically by <code>ldapControlAgent</code> (on page 33).</td>
</tr>
<tr>
<td>requestingSystem</td>
<td>All Types</td>
<td>Up to 14 hex digits</td>
<td>The MSID (mobile subscriber ID).</td>
</tr>
<tr>
<td>merchantID</td>
<td>All Types</td>
<td>Up to 16 characters</td>
<td>An identifier for the provider of the SMS message. This is not used specifically by <code>ldapControlAgent</code>.</td>
</tr>
<tr>
<td>service</td>
<td>Debit</td>
<td>Up to 6 characters</td>
<td>This indicates the type of service – &quot;MO&quot;, &quot;MT&quot;, &quot;MO_EMS&quot; or &quot;MT_EMS&quot;. This is used by <code>ldapControlAgent</code> to identify the IDP mappings to use (to distinguish between debitMO and debitMT).</td>
</tr>
<tr>
<td>content</td>
<td>Debit</td>
<td>Up to 16 characters</td>
<td>A string which describes the nature of the message content (for example: &quot;EMAIL&quot;).</td>
</tr>
<tr>
<td>units</td>
<td>Debit</td>
<td>Decimal string</td>
<td>This indicates the number of messages to debit for. Usually &quot;1&quot;.</td>
</tr>
<tr>
<td>balanceIndicator</td>
<td>Debit</td>
<td>Decimal string</td>
<td>This indicates the balance to debit. This is always expected to be &quot;0&quot;.</td>
</tr>
<tr>
<td>debitType</td>
<td>Debit</td>
<td>Up to 3 characters</td>
<td>The type of debit. While <code>ldapControlAgent</code> does not specifically check this field, the only expected value is &quot;300&quot;.</td>
</tr>
<tr>
<td>destinationAddress</td>
<td>Debit</td>
<td>Up to 16 hex digits</td>
<td>The &quot;other party&quot;. For SMS-MO this is the destination address.</td>
</tr>
</tbody>
</table>
## Field | Used by | Format | Note
--- | --- | --- | ---
transactionAmount | Credit | Decimal | The amount of the credit, in the specified currency (for example: "12.5000").
currencyLabel | Credit | Up to 24 characters | Label for the currency. It is expected to be text such as "USD". For credit, the currency of the credited amount. For balance, the currency in which the returned balance should be presented.
creditType | Credit | Up to 3 characters | Type/reason of credit. While ldapControlAgent does not specifically check this field, the only expected value is "100".
internationalIndicator | Debit | One decimal | Set to "1" if any of the numbers in the connectedDNlist are international. Otherwise this is empty.
connectedDNlist | Debit | Hex digits + a single comma | This is a comma-separated list of destination addresses.

**Note:** Most fields are not used directly by ldapControlAgent, and are passed to slee_acs without interpretation. There is one specific exception to that rule. Field 6 "service" in a Debit message is interpreted by ldapControlAgent. It must begin with either "MO" or "MT". It is used to select the appropriate mapping entry – either "debitMO" or "debitMT". This is hard-coded in ldapControlAgent.

### sid filter field mapping

For flexibility, LCA is configurable in terms of which SearchRequest "sid" filter string fields ldapControlAgent maps to which InitialDP parameters. This mapping is done on a per-service-type basis, with the following service types available:

- debitMO
- debitMT
- credit
- balance

For debit messages ldapControlAgent will look specifically at field index 6 to identify whether the debit is a MO or MT debit. Once this decision is made, the appropriate section of the idpMapping configuration is used.

### InitialDP fields

The following INAP InitialDP fields can be populated from the incoming LDAP message. Most items specify the sid filter attribute colon-separated index field number to use as the source. The eventTypeBCSM is the exception, it is a specific integer value.

<table>
<thead>
<tr>
<th>IDP Field</th>
<th>eserv.config parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>calledPartyNumber</td>
<td>calledPartyNumberField (on page 27)</td>
<td>zero-based field Index</td>
</tr>
<tr>
<td>callingPartyNumber</td>
<td>callingPartyNumberField (on page 27)</td>
<td>zero-based field Index</td>
</tr>
</tbody>
</table>
sid filter field parameters

ldapControlAgent supports the following parameters in the `idpMapping` subsection of `eserv.config`.

**Note:** ldapControlAgent also supports service key configuration in the `idpMapping` subsection. For more information about service key configuration, see Service Key Configuration (on page 22).

### calledPartyNumberField

**Syntax:**

```
calledPartyNumberField = int
```

**Description:**
The location in the sid filter field to use as the source for the `calledPartyNumberField` field in the InitialDP.

**Type:**
Integer

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

**Allowed:**

**Default:**
-1 (do not populate)

**Notes:**

**Example:**

### callingPartyNumberField

**Syntax:**

```
callingPartyNumberField = int
```

**Description:**
The location in the sid filter field to use as the source for the `callingPartyNumberField` field in the InitialDP.

**Type:**
Integer

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

**Allowed:**

**Default:**
-1 (do not populate)

**Notes:**

**Example:**

### locationNumberField

**Syntax:**

```
locationNumberField = int
```

**Description:**
The location in the sid filter field to use as the source for the `locationNumberField` field in the InitialDP.

**Type:**
Integer

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

**Allowed:**

---

<table>
<thead>
<tr>
<th>IDP Field</th>
<th>eserv.config parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>locationNumber</td>
<td><code>locationNumberField</code> (on page 27)</td>
<td>zero-based field Index</td>
</tr>
<tr>
<td>originalCalledPartyID</td>
<td><code>originalCalledPartyIDField</code> (on page 28)</td>
<td>zero-based field Index</td>
</tr>
<tr>
<td>redirectingPartyID</td>
<td><code>redirectingPartyIDField</code> (on page 28)</td>
<td>zero-based field Index</td>
</tr>
<tr>
<td>additionalCallingPartyNumber</td>
<td><code>additionalCallingPartyNumberField</code> (on page 28)</td>
<td>zero-based field Index</td>
</tr>
<tr>
<td>eventTypeBCSM</td>
<td><code>eventTypeBCSM</code> (on page 28)</td>
<td>INTEGER VALUE</td>
</tr>
</tbody>
</table>

---
Default: -1 (do not populate)

Notes:

Example:

**originalCalledPartyIDField**

Syntax: `originalCalledPartyIDField = int`

Description: The location in the sid filter field to use as the source for the `originalCalledPartyIDField` field in the InitialDP.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: Default: -1 (do not populate)

Notes: Example:

**redirectingPartyIDField**

Syntax: `redirectingPartyIDField = int`

Description: The location in the sid filter field to use as the source for the `redirectingPartyIDField` field in the InitialDP.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: Default: -1 (do not populate)

Notes: Example:

**additionalCallingPartyNumberField**

Syntax: `additionalCallingPartyNumber = int`

Description: The location in the sid filter field to use as the source for the `additionalCallingPartyNumberField` field in the InitialDP.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: Default: -1 (do not populate)

Notes: Example:

**eventTypeBCSM**

Syntax: `eventTypeBCSM = int`

Description: The value to set `eventTypeBCSM` to in the message to the SMSC.

Type: Integer

Optionality: Mandatory

Allowed: Default: No default

Notes: `slee_acs` may interpret these BCSM state numbers and modify its call control behavior accordingly.
Examples:

```
eventTypeBCSM = 3
eventTypeBCSM = 12
```

**Extension tags**

In addition to the standard InitialDP fields listed above, ldapControlAgent can also pass any field through to slee_acs in a special “extensions” field provided by the INAP InitialDP. This “extensions” field is encoded as a profile block which is then made available in ACS control plans via the profile block “Incoming Extensions”.

Extension tags are listed in the following format on a per service type basis:

```
extensionTags = [  
    { source = 6, type = "integer", scale = 100, tag = 7077890 }  
    { source = 7, type = "string", tag = 7077891 }  
]```

This mechanism is similar to the configuration for placing “sid” filter fields into standard INAP fields, with the following exceptions.

The standard INAP address fields allow only 0-0, A-F. Extension “string” fields allow any ASCII character (except the colon “:” which is reserved for field delimiters in TCS LDAP).

The extension “integer” type allows for values to be scaled before they are written in to profile integer fields. This is useful since TCS LDAP provides credit amounts in “big money” (for example: “12.4500” for $12.45). slee_acs requires integer “small money” amounts (for example: 1245 for $12.45).

**Extension tag parameters**

Each extension tag parameter array has the following configuration elements.

- **source**
  - **Syntax:** `source = int`
  - **Description:** The source field in the sid “filter” string provided by the incoming LDAP message.
  - **Type:** Integer
  - **Optionality:** Mandatory
  - **Allowed:**
    - **Default:** No default
  - **Notes:** This is zero-based (that is, the first item is specified by 0, not 1).
  - **Example:** `source = 6`

- **type**
  - **Syntax:** `type = "type"`
  - **Description:** The type of data that is being passed through the profile block.
  - **Type:** String
  - **Optionality:** Mandatory
  - **Allowed:**
    - **Allowed values are:**
      - “string”
      - “integer”
  - **Default:** No default
  - **Notes:**
  - **Example:** `type = "string"`
scale

Syntax: \[ \text{scale} = \text{int} \]

Description: Amount to scale integers during the LDAP to INAP translation.

Type: Integer

Optionality: Mandatory if \( \text{type} \) (on page 29) = "integer"

Allowed: No default

Default: No default

Notes: Some integers being passed to slee_acs from LdapControlAgent may be balances. In this case, they need to be scaled in the opposite scale to the scale we use when converting balances from slee_acs.

Example: \( \text{scale} = 100 \)

tag

Syntax: \[ \text{tag} = \text{tag} \]

Description: The tag field defines the profile block tag to store the source data in.

Type: Integer

Optionality: Mandatory

Allowed: No default

Default: No default

Notes: ACS must be configured to look in the appropriate tag field. For more information about LCA profile tags, see Profile tags (on page 2).

Example: \( \text{tag} = 7077891 \)

Returned Parameter Mapping

Introduction

In the case of "success" or "soft error", LdapControlAgent will return a SearchResponseEntry. Both INAP Connect and ReleaseCall responses from slee_acs will result in SearchResponseEntry messages to the SMSC.

LdapControlAgent will ensure the following basic values are echoed back from the original message:

- MessageID (copy the messageID from the original message)
- objectName (copy the "filter" string from the original message)

In addition, LdapControlAgent will set one or more attribute fields.

- Connect (success) will specify attribute "res" = "00", plus additional attributes.
- ReleaseCall (soft error) will specify attribute "res" != "00". No other attributes.

The number and nature of each depend on the type of the original request.

Response attributes by request type

This table describes the function of each field.

<table>
<thead>
<tr>
<th>Attr.</th>
<th>Debit (MO + MT)</th>
<th>Credit</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>bal</td>
<td>Resultant balance. ASCII fractional decimal (for example &quot;14.3000&quot;). Taken from integer profile field in Connect profile block. Located through tag LCA.balanceProfileTag (on page 17). Multiplied by LCA.balanceScale (on page 17), formated with LCA.balancePrecision (on page 18) places after decimal point.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attr.</td>
<td>Debit (MO + MT)</td>
<td>Credit</td>
<td>Balance</td>
</tr>
<tr>
<td>-------</td>
<td>----------------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>bal1</td>
<td></td>
<td></td>
<td>Current balance. See &quot;bal&quot;.</td>
</tr>
<tr>
<td>bt1</td>
<td></td>
<td></td>
<td>LCA.balanceCode (on page 17)</td>
</tr>
<tr>
<td>bal2</td>
<td></td>
<td></td>
<td>&quot;&quot; (empty)</td>
</tr>
<tr>
<td>bt2</td>
<td></td>
<td></td>
<td>&quot;&quot; (empty)</td>
</tr>
<tr>
<td>cur</td>
<td></td>
<td></td>
<td>Set to global-configured ASCII currency string defined by LCA.currencyCode (on page 16).</td>
</tr>
<tr>
<td>res</td>
<td></td>
<td></td>
<td>Set to two-character ASCII decimal field. If slee_acs sent a Connect, this will be mapped to &quot;00&quot;. Otherwise the value is defined by ReleaseCall cause mapping defined in the LCA.releaseMap (on page 31) configuration.</td>
</tr>
<tr>
<td>amt</td>
<td></td>
<td></td>
<td>Not returned by LCA.</td>
</tr>
<tr>
<td>inov</td>
<td></td>
<td></td>
<td>Not returned by LCA.</td>
</tr>
</tbody>
</table>

**ReleaseCall mapping configuration**

By default, a ReleaseCall will result in a "res" attribute != "00" and no other attributes. The release cause mapping configuration is used to configure the "res" value eventually delivered to the SMSC.

```plaintext
LCA = {
    releaseMap = [
        { cause = int, result = code }
        ...
    ]
    defaultReleaseCallResult = code
}
releaseMap
```

**Syntax:**

```plaintext
releaseMap [  
    { cause = int, result = code }
    ...
]  
```

**Description:**
The mapping from ReleaseCall to ReleaseCode.

**Type:**
Array

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

**Allowed:**

**Default:**
`defaultReleaseCauseResult` (on page 19)

**Notes:**
More than one cause may map to the same result code. While this configuration is in integers, the return format is a string. cause and result are mandatory if this array is used.

**Example:**
For an example of this parameter used in context, see Example eserv.config file (on page 13).
LCA Control Plan

Introduction

In practice, the LCA functionality will interact with a ACS control plan. The control plan provides the service logic the LCA is providing access to. For example, a control plan using CCS nodes would provide access to subscriber balance information.

Minimum control plan requirements

To indicate success to ldapControlAgent, the control plan needs to:

- Set the “LCA Subscriber Balance” profile field in the “Outgoing Extensions” profile to be an integer value representing the “small money” subscriber balance
- Use a feature node which returns a Connect message (for example: Unconditional Termination)

To indicate failure to ldapControlAgent, a control plan needs to use a feature node which produces a ReleaseCall (for example: Disconnect).

For more information about:

- LCA profile fields, see Profile tags (on page 2).
- Feature nodes and setting up control plans, see CPE User's Guide.
- CCS feature nodes, see CCS Feature Node User's Guide.

Example control plan

Here is an example success control plan, suitable for testing.
Overview

Introduction

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

- inittab
- crontab
- Service Logic Execution Environment SLEE

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

In this chapter

This chapter contains the following topics.

IdapControlAgent

IdapControlAgent

Purpose

IdapControlAgent is the main process in the Oracle Communications Network Charging and Control LDAP Control Agent (LCA) component.

It is a SLEE interface process. It listens on one or more TCP port numbers and accepts LDAP connections. It uses the SLEE to send INAP messages to any SLEE application which can accept an InitialDP and return a Connect message with the necessary extension parameters set (for example: slee_acs).

Location

This binary is located on SLCs.

Startup

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

```
INTERFACE=lcaIf lca.sh /IN/service_packages/LCA/bin EVENT
SERVICE=LCA_General 1 slee_acs LCA_General
SERVICEKEY=INTEGER 108001 LCA_General
```

Notes:

- lca.sh is a shell script which starts the IdapControlAgent process.
- The above are defaults and may vary.
In a standard configuration, only a single instance of the ldapControlAgent will be started. Multiple instances of the ldapControlAgent interface process can be run (for improved performance on a multi-CPU box if the ldapControlAgent appears to be CPU-limited). Each ldapControlAgent instance needs to use a separate eserv.config file so the IP address and TCP Port Numbers do not conflict. This can be done by configuring each instance to use a setting for the environment variable ESERV_CONFIG_FILE which points to a different eserv.config file.

**Configuration**

In order to load and operate, the ldapControlAgent reads the LCA section of the eserv.config file. For more information about the LCA section and ldapControlAgent configuration, see *Configuration* (on page 11).

For more information about causing ldapControlAgent to reread its configuration, see *Loading eserv.config configuration changes* (on page 13).

**Example configuration**

For an example of the LCA section of a eserv.config file, see *Example eserv.config file* (on page 13).

**Failure**

The ldapControlAgent will be monitored by the SLEE watchdog. The watchdog will restart ldapControlAgent if it fails to respond to regular heartbeat events. For more details about how the watchdog monitors SLEE processes, see *SLEE Technical Guide*.

ldapControlAgent generates standard SMS alarm log messages to alert operators to any misconfiguration or abnormal processing. For more information about the alarms generated by ldapControlAgent, see *LCA Alarms Guide*.

**Output**

ldapControlAgent writes error messages to the system messages file. Under normal processing it will echo all alarm messages to STDERR. As a SLEE process, this will be merged with all SLEE output.

For information about debug output, see *Introduction*.

**Status reports**

At any time, you can send ldapControlAgent a SIGUSR1 which causes it to print a dump of its:

- Current configuration
- List of connected servers
- Statistics on each connected server
- Examples of outstanding SMS requests yet to be responded to

This summary will be printed to the standard error of the process. The actual destination for standard error is site-specific, and will depend on how your SLEE process has been started and how it has been configured.

Generating the status report will not affect the connections.
About Installation and Removal

Overview

Introduction

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

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation and Removal</td>
<td>35</td>
</tr>
<tr>
<td>Checking the Installation</td>
<td>35</td>
</tr>
<tr>
<td>Post-installation Configuration</td>
<td>37</td>
</tr>
</tbody>
</table>

Installation and Removal Overview

Introduction

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

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

LCA packages

An installation of Oracle Communications Network Charging and Control LDAP Control Agent includes the following packages, on the:

- SMS:
  - lcaSms
- SLC:
  - lcaScp

Checking the Installation

Introduction

The checklists in this section should be used to ensure that the LCA component has installed correctly.

Checking the lcaSms installation

Follow these steps to check whether the LCA has installed onto a SMS correctly.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log in to the SMS machine as root.</td>
</tr>
</tbody>
</table>
Chapter 4

Step | Action
--- | ---
2 | Check the following directory structure exists with subdirectories:
   /IN/service_packages/LCA
3 | Check that directories contain subdirectories and that all are owned by smf_oper user (group esg).
4 | Log into the SMF database as the user SMF.
5 | Check the following SQL checks.

Checking the SMF installation

This table lists the SQL commands which should be run to check that the lcaSms package installed the SMF database changes correctly.

<table>
<thead>
<tr>
<th>Check</th>
<th>Expected output</th>
<th>Confirms</th>
</tr>
</thead>
<tbody>
<tr>
<td>select count(*) from smf_statistics_defn where application_id = 'LCA';</td>
<td>COUNT(*)</td>
<td>12 statistics configured for the LCA.</td>
</tr>
<tr>
<td>select count(*) from acs_profile_details where profile_tag &gt;= 7077888 and profile_tag &lt;= 7117209;</td>
<td>COUNT(*)</td>
<td>4 different profile tags unique to the LCA.</td>
</tr>
<tr>
<td>select count(*) from acs_tag_to_profile_mapping where profile_tag &gt;= 7077888 and profile_tag &lt;= 7117209;</td>
<td>COUNT(*)</td>
<td>All 4 different profile tags included in ACS profile tag mappings so that they are available in the ACS Control Plan Editor.</td>
</tr>
</tbody>
</table>

Checking the lcaScp installation

Follow these steps to check whether the LCA has installed onto a SLC correctly.

Step | Action
--- | ---
1 | Log in to the SLC machine as root.
2 | Check the following directory structure exists with subdirectories:
   /IN/service_packages/LCA
3 | Check that directories contain subdirectories and that all are owned by acs_oper user (group esg).
4 | Check that the LCA configuration has been installed into:
   /IN/service_packages/eserv.config
5 | Check that the expected processes are running.
   For a list of the processes which should be running, see Process list (on page 36).

Process list

If LCA is running, the following processes should be running on the SLC

- Started during SLEE startup:
  - ldapControlAgent (on page 33)

Note: LCA does not run any processes on the SMS.
Post-installation Configuration

Configuration process overview

This table describes the steps involved in configuring LCA for the first time.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Uncomment (and if necessary update) the default SLEE configuration for the LCA in the <code>SLEE.cfg</code> file. For more information, see <em>SLEE Technical Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>The <code>eserv.config</code> file must be configured for LCA. The installation script will have added an LCA section to <code>eserv.config</code>. Any required configuration should be updated. For more information, see <em>eserv.config Configuration</em> (on page 12).</td>
</tr>
<tr>
<td>3</td>
<td>Configure the statistics replication to all SLCs which are running LCA. For more information about how to set up replication, see <em>Setting up statistics replication</em> (on page 37).</td>
</tr>
</tbody>
</table>

Setting up statistics replication

Follow these steps to configure SMS replication to ensure all LCA statistics definitions are available on all SLC nodes which are running LCA.

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

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Restart SMS Java (Swing) Administration screens.</td>
</tr>
<tr>
<td>2</td>
<td>Open the <em>Table Replication</em> tab on the SMS Node Management screen.</td>
</tr>
<tr>
<td>3</td>
<td>LCA installation should have automatically configured a <code>SMS.SMF_STATISTIC_DEFN.SMF_STDEF_LCA</code> entry to the left hand panel of the screen. Add the new entry to the SLC nodes which have LCA installed.</td>
</tr>
<tr>
<td>4</td>
<td>Save the updated node config, by clicking <em>Save</em>.</td>
</tr>
<tr>
<td>5</td>
<td>Click <em>Create Config File</em> update the SLCs.</td>
</tr>
<tr>
<td>6</td>
<td>Send a SIGHUP signal to all <code>smsStatsDaemon</code> nodes so that the added LCA statistics are recognized when they are generated by <em>ldapControlAgent</em> (on page 33).</td>
</tr>
</tbody>
</table>
NCC Glossary of Terms

ACS
Advanced Control Services configuration platform.

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

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

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

CPU
Central Processing Unit

cron
Unix utility for scheduling tasks.

crontab
File used by cron.

DP
Detection Point

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

EDR
Event Detail Record

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

ETSI
European Telecommunications Standards Institute
**HTML**

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

**IDP**

INAP message: Initial DP (Initial Detection Point)

**IN**

Intelligent Network

**INAP**

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

**Initial DP**

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

**IP**

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

**IP address**

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

**MDN**

Mobile Directory Number

**MIN**

Mobile Identification Number, also known as an MSID.

**MO**

Mobile Originated

**MS**

Mobile Station

**MSID**

Mobile Subscriber Identification, also known as an MIN.

**MT**

Mobile Terminated
Oracle
Oracle Corporation

SGML

SLC
Service Logic Controller (formerly UAS).

SLEE
Service Logic Execution Environment

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

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

SMS-MO
Short Message Service Mobile Originated

SMS-MT
Short Message Service Mobile Terminating

SNMP

SQL
Structured Query Language - a database query language.

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

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

VWS
Oracle Voucher and Wallet Server (formerly UBE).