

# CUDB LDAP Interwork Description

---

## INTERWORK DESCRIPTION

**Copyright**

© Ericsson AB 2016, 2017. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

**Disclaimer**

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

**Trademark List**

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Scope	1
1.2	Revision Information	1
1.3	Target Groups	2
1.4	Typographic Conventions	2
<b>2</b>	<b>Overview</b>	<b>3</b>
<b>3</b>	<b>CUDB Data Model Description</b>	<b>5</b>
3.1	CUDB Main Directory Information Tree	5
3.2	CUDB Application Counters Directory Information Tree	7
3.3	CUDB Root Entry	8
3.4	CUDB LDAP Users and Groups	8
3.5	Identity Data	9
3.6	Multi Service Consumer and Association Data	12
3.7	MSC Common Data	14
3.8	Service Common Data	15
3.9	LDAP Views	16
<b>4</b>	<b>Attributes and Object Classes</b>	<b>19</b>
4.1	Attributes	19
4.2	Object Classes	22
4.3	Operational Attributes	22
4.4	Limitations	23
<b>5</b>	<b>CUDB DIT Entries</b>	<b>25</b>
5.1	CUDB Admin Entry	25
5.2	Identities Container Entry	25
5.3	Multi Service Consumer Container Entry	25
5.4	Associations Container Entry	26
5.5	Multi Service Consumer Common Data Container Entry	27
5.6	Service Common Data Container Entry	28
<b>6</b>	<b>Operations</b>	<b>31</b>
6.1	CUDB Shortcut in Search for Authentication Data	33



6.2	Performing LDAP Search Below the Distributed Entry	34
6.3	Error Codes	34
6.4	LDAP Controls	38
	<b>Glossary</b>	<b>41</b>
	<b>Reference List</b>	<b>43</b>



# 1 Introduction

This document describes the Ericsson reference data model for the basic and initial data maintained in the Ericsson Centralized User Database (CUDB). This document provides a detailed description of Lightweight Directory Access Protocol (LDAP) objects, classes, and directory entries including attribute description, format, allowed value range, and the allowed LDAP operations.

## 1.1 Scope

This document covers the following:

- CUDB data model description.
- Attributes and objects classes.
- CUDB DIT entries.
- Operations to handle CUDB data.

The description of any other application Front End (FE) specific data or internal CUDB data is out of the scope of this document.

## 1.2 Revision Information

### Rev. A

This document is based on 1/15519-HDA 104 03/9 with the following changes:

- Section 3.4 on page 8, Section 3.9 on page 16, and Section 6 on page 31: Updated information on function availability.
- Section 3.6 on page 11: Added information about container entries.
- Section 6.3 on page 34: Added error code 3 to Table 16 and updated proxy timeout information for error code 11.

### Rev. B

Other than editorial changes, this document has been revised as follows:

- Section 6.4 on page 38: New section.



## 1.3 Target Groups

This document is intended for users working with CUDB LDAP schemas. This document assumes the general knowledge of the LDAP standard.

## 1.4 Typographic Conventions

Typographic conventions can be found in the following document:

- *Typographic Conventions*



## 2 Overview

CUDB is the network entity in a layered architecture domain serving as central storage point for application FE data (Home Location Register FE or HLR-FE data, IMS data, and so on).

CUDB is built as LDAP directory server, containing the needed entries and attributes according to the defined schema for the different application FEs.

LDAP is a client-server based directory access protocol that provides both read and update access.

LDAP data information model provides the structures and data types necessary for building an LDAP directory tree.

This document describes the LDAP data provided in the following LDAP schema:

- `/cluster/home/cudb/dataAccess/ldapAccess/ldapFe/config/schema/cudb.schema`
- `/cluster/home/cudb/dataAccess/ldapAccess/ldapFe/config/schema/application_counters.schema`







## 3 CUDB Data Model Description

The following sections describe CUDB data model. For more information on LDAP data model, refer to *CUDB LDAP Data Access*, Reference [1].

### 3.1 CUDB Main Directory Information Tree

From LDAP modelling perspective in CUDB, the Directory Information Tree (DIT) is built into the following structures:

- CUDB root entry.
- Administration LDAP user data (out of the scope of this document).
- Identity data.
- Multi service consumer data.
- Association data.
- Multi service consumer common data.
- Service common data.

Figure 1 shows CUDB LDAP DIT.

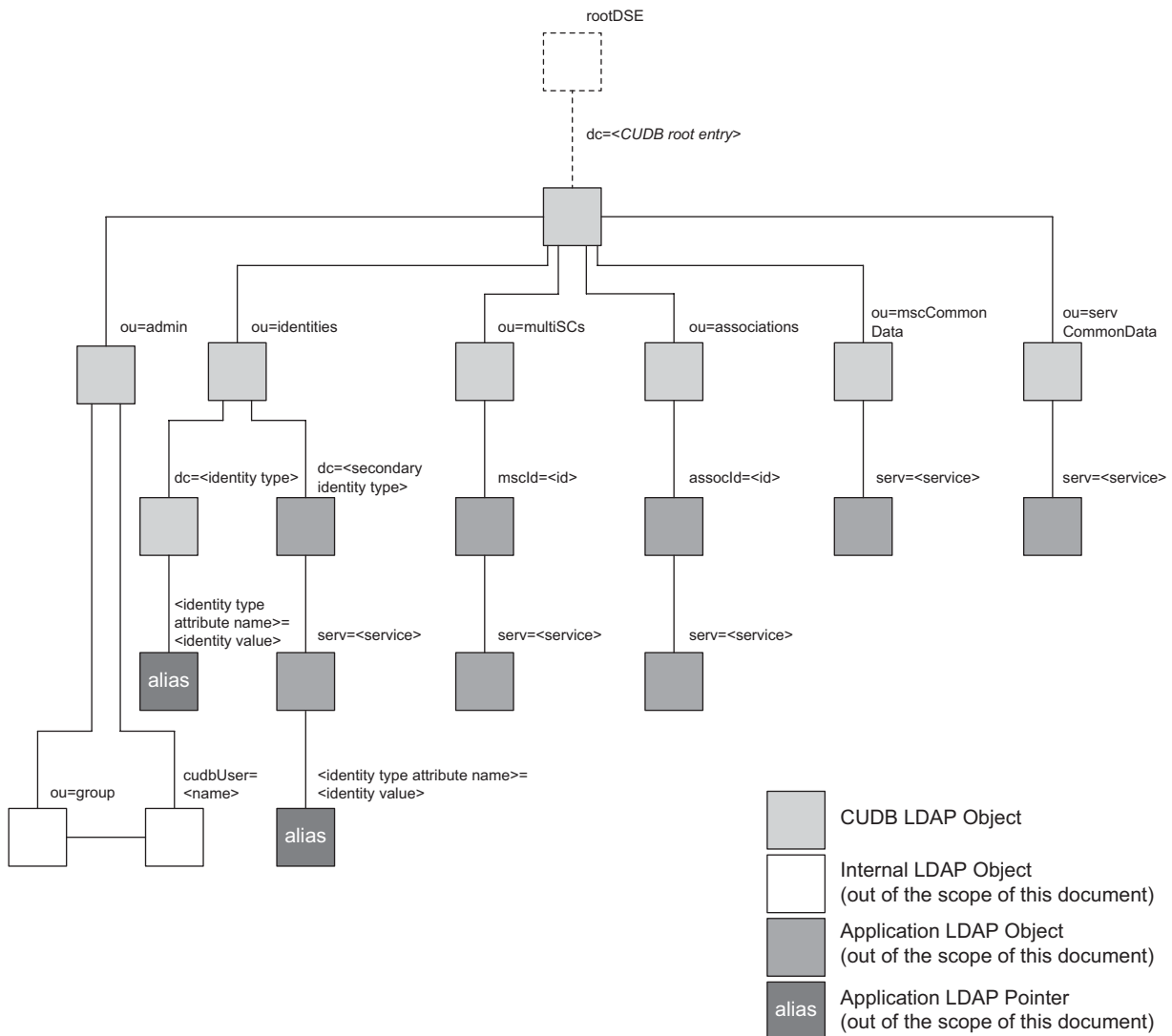


Figure 1 CUDB LDAP DIT

CUDB has two built-in distribution entries: MSCs and Associations.

CUDB also defines custom distribution entries. Custom of distribution entries must fulfill the following requirement:

- When a distribution entry is provisioned, the following two optional attributes must be defined in any of the object classes it includes:
  - ZoneId.
  - DSUnitGroup.

For more information, refer to *CUDB Node Configuration Data Model Description*, Reference [3].



## 3.2 CUDB Application Counters Directory Information Tree

CUDB provides read-only access through the LDAP interface to the set of Application Counters (for more information, refer to *CUDB Application Counters, Reference* [4]) by means of exposing those counters in a separate LDAP DIT:

- Root Entry (ou=ApplicationCounter)
- Application Counters Group entries (dn: cn=<Counters Group Name>, ou=ApplicationCounter). These entries host no data
- Application Counters entries (dn: cn=<Application Counter Name>, cn=<Counters Group Name>, ou=ApplicationCounter) organized below their respective Counters Group parent entry, and hosting the Counters name and value.

Figure 2 shows the CUDB Licensing DIT.

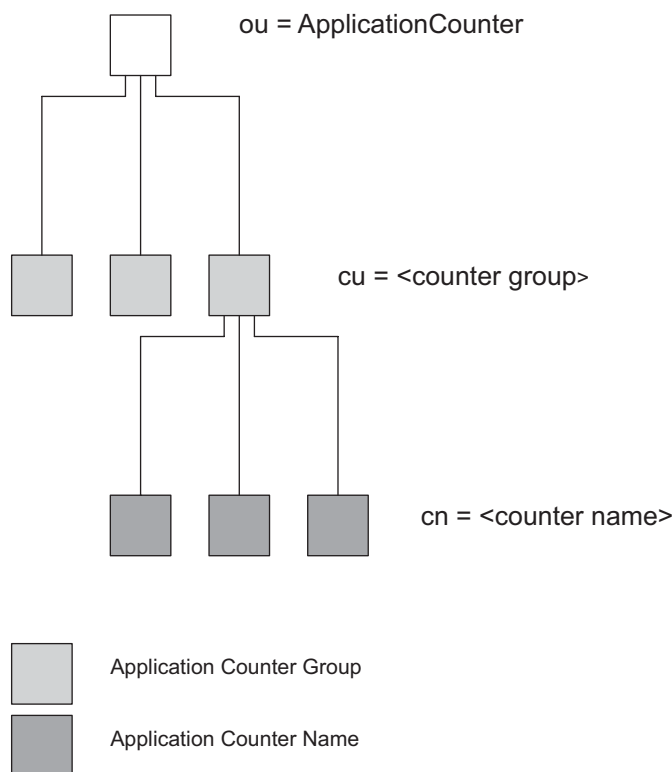


Figure 2 CUDB Licensing DIT

The new backend of the LDAP FE for counters fetches the value of the column *<Application Counter Name>* from the table *<Counters Group Name>*, and puts it in the generic 'value' LDAP attribute. To fetch a counter value, the following LDAP search operation is used:



- Basedn: "ou=ApplicationCounter, cn=<CountersGroupId>, cn=<CounterId>"
- Scope: any (base/one/subtree will return the same)
- Filter: objectclass=\* or cn=<CounterId> or CounterValue=\* can be used

An example of an LDAP search operation is shown below:

```
# ldapsearch -h pl_2_3 -D "cn=manager,dc=operator,dc=com" -w normal -b "cn=NSUBSCNT,cn=GRP_HLRSUBS,ou=ApplicationCounter" -s base # extended LDIF
#
# LDAPv3
# base <cn=NSUBSCNT,cn=GRP_HLRSUBS,ou=ApplicationCounter> with scope baseObject
# filter: (objectclass=*)
# requesting: ALL
#
# NSUBSCNT, GRP_HLRSUBS, ApplicationCounter
dn: cn=NSUBSCNT,cn=GRP_HLRSUBS,ou=ApplicationCounter
objectClass: ApplicationCounter
cn: NSUBSCNT
CounterValue: 3924600
# search result
search: 2
result: 0 Success
# numResponses: 2
# numEntries: 1
```

### 3.3 CUDB Root Entry

The Distinguished Name (DN) and Relative Distinguished Name (RDN) of the CUDB root entry are fully configurable at CUDB installation time.

### 3.4 CUDB LDAP Users and Groups

For application FEs to access the data stored in CUDB via the LDAP interface, one or more LDAP users must be defined with the corresponding credentials and configuration parameters. Application FEs may bind to the CUDB LDAP interface with one of these users and then send the corresponding LDAP operations to CUDB. Groups of users may optionally be defined to share access control rules between LDAP users. For more information on how LDAP users and LDAP users groups are defined in CUDB, refer to *CUDB System Administrator Guide*, Reference [5].

Due to the LDAP Data Views function, LDAP users that have an LDAP data view assigned to them can access the data through a custom DIT. Users without an assigned LDAP data view can access the data through the core DIT.

**Note:** The LDAP Data Views function can only be used if the Application Facilitator Value Package is available.



CUDB LDAP users and groups are defined through configuration. For more information, refer to *CUDB Node Configuration Data Model Description*, Reference [3].

Following are the administration data entries defined in CUDB:

- Admin container, "ou=admin,<CUDB root entry>"
- LDAP User data, "cudbUser=<CUDB LDAP user>,ou=admin,<CUDB root entry>"
- LDAP Group data, "ou=<CUDB LDAP group>,ou=admin,<CUDBroot entry>"
- LDAP User data belonging to a LDAP Group, "cudbUser=<CUDB LDAP user>,ou=<CUDB LDAP group>,ou=admin,<CUDBroot entry>"

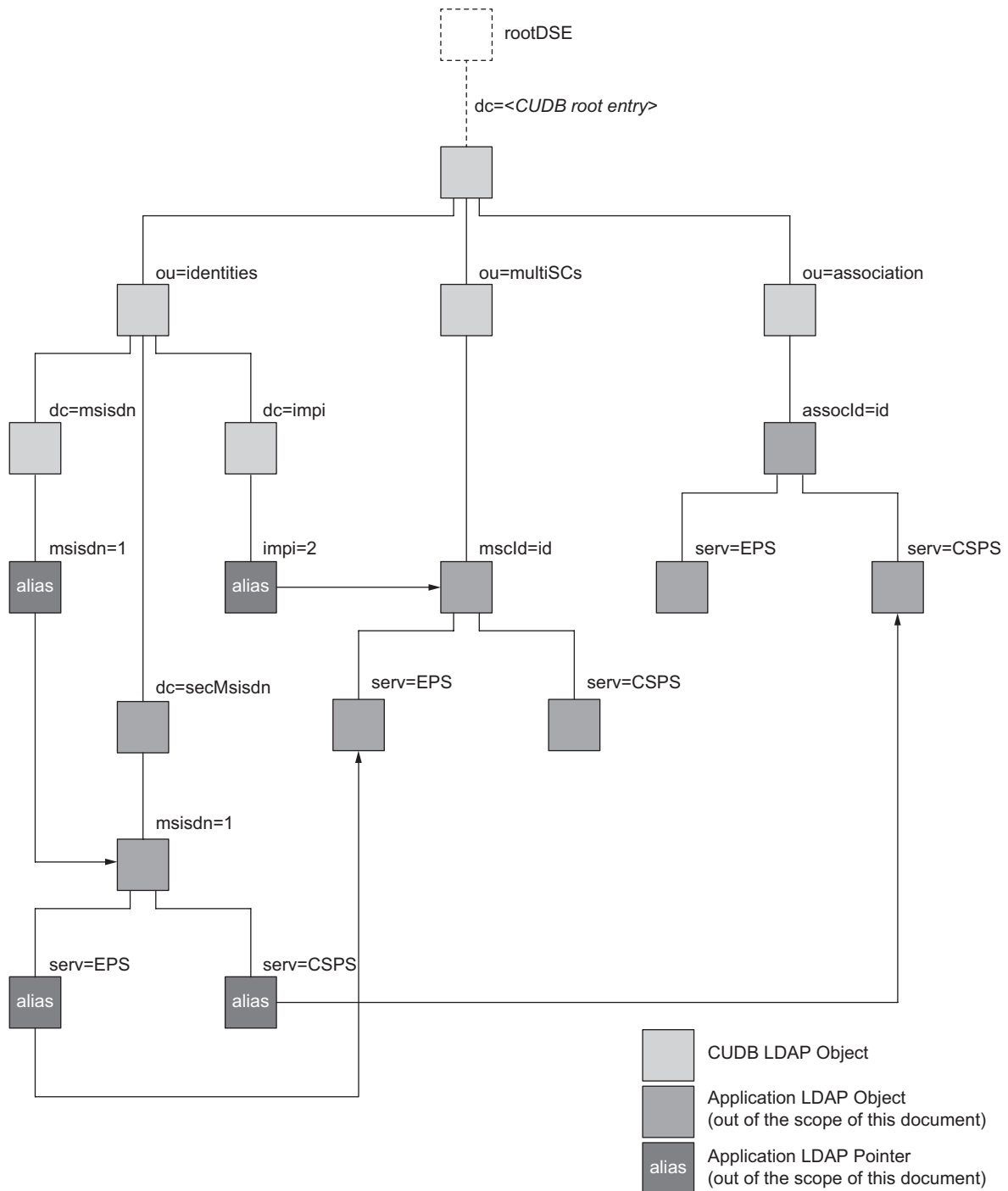
## 3.5 Identity Data

The different identities used to identify and access subscribers (typically MSC or Association) data are placed under ou="identities" container level entry.

Following are two types of identity entries depending on how these entries are used as reference to an MSC or an Association:

- Primary identities: These entries are always aliases to another entry (typically to an MSC, an Association, or to a secondary identity). As primary identities serve this specific purpose, they are treated as special entries in CUDB in the sense that each type must be defined at installation time and entries are stored in a particular way to optimize the space taken by each entry. Besides this, primary identities can only be stored right under its type container which is populated at installation time. Moreover, CUDB supports a non standard behavior of the LDAP modify operations (alias de-referencing) when the entry to be modified contains a primary identity. For more details, see Section 6 on page 31.
- Secondary identities: These entries are similar to primary identities in the sense that they are usually used to have references to other entries but the alias does not have to be in the identity itself but also below it. Unlike primary identities, these identities are not treated as special entries in CUDB. For example, a secondary identity could be an entry that contains subentries for different services where each subentry contains an alias referencing to a different MSC or Association.

Figure 3 provides an example of these primary and secondary identity entries.



**Figure 3 Primary and Secondary Identity Entries**

Following are primary identity data entries defined in CUDB:

- Identity container, "ou = identities, <CUDBroot entry>"



Different containers for different types of identities might be defined related to identity data:

- Identity domain container, "dc=<identity type>, ou=identities, <CUDB root entry>"

<identity type> can take any value. For example, msisdn, imsi, impi, impu, ipaddress,...

The entry must contain at least an LDAP object class containing the attribute dc. For example, dcObject.

Following is an example of container for MSISDN:

```
"dc = msisdn, ou = identities, <CUDB root entry>"
```

Under each different container for the different types of identities alias entries per identity might be defined:

- Identity alias, "<identity type attribute name>=<identity value>, dc=<identity type>, ou = identities, <CUDB root entry>"

<identity type attribute name> can take any name. For example, msisdn, imsi, impi, impu, ipaddress,...

This attribute, its type and syntax and the LDAP object class are defined and provided in an LDAP schema by CUDb LDAP clients. The only condition is that the object class must be of type auxiliary.

The entry must contain the 'alias' object class containing the attribute aliasedObjectName. For example, alias.

**Note:** As the 'alias' object class is a structural class, the <identity type attribute> can be added using the 'extensibleObject' object class. Auxiliary object class which, when present in an entry, permits the entry to optionally hold any attribute. The allowed attribute list of this class is implicitly the set of all attributes known to the server.

Following is an example of MSISDN alias:

```
"MSISDN =<MSISDNnumber>, dc = msisdn, ou = identities, <CUDB root entry>"
```

The identity type attribute name must be the same as the identity type. In the above example, the identity type attribute name MSISDN matches the identity type msisdn.



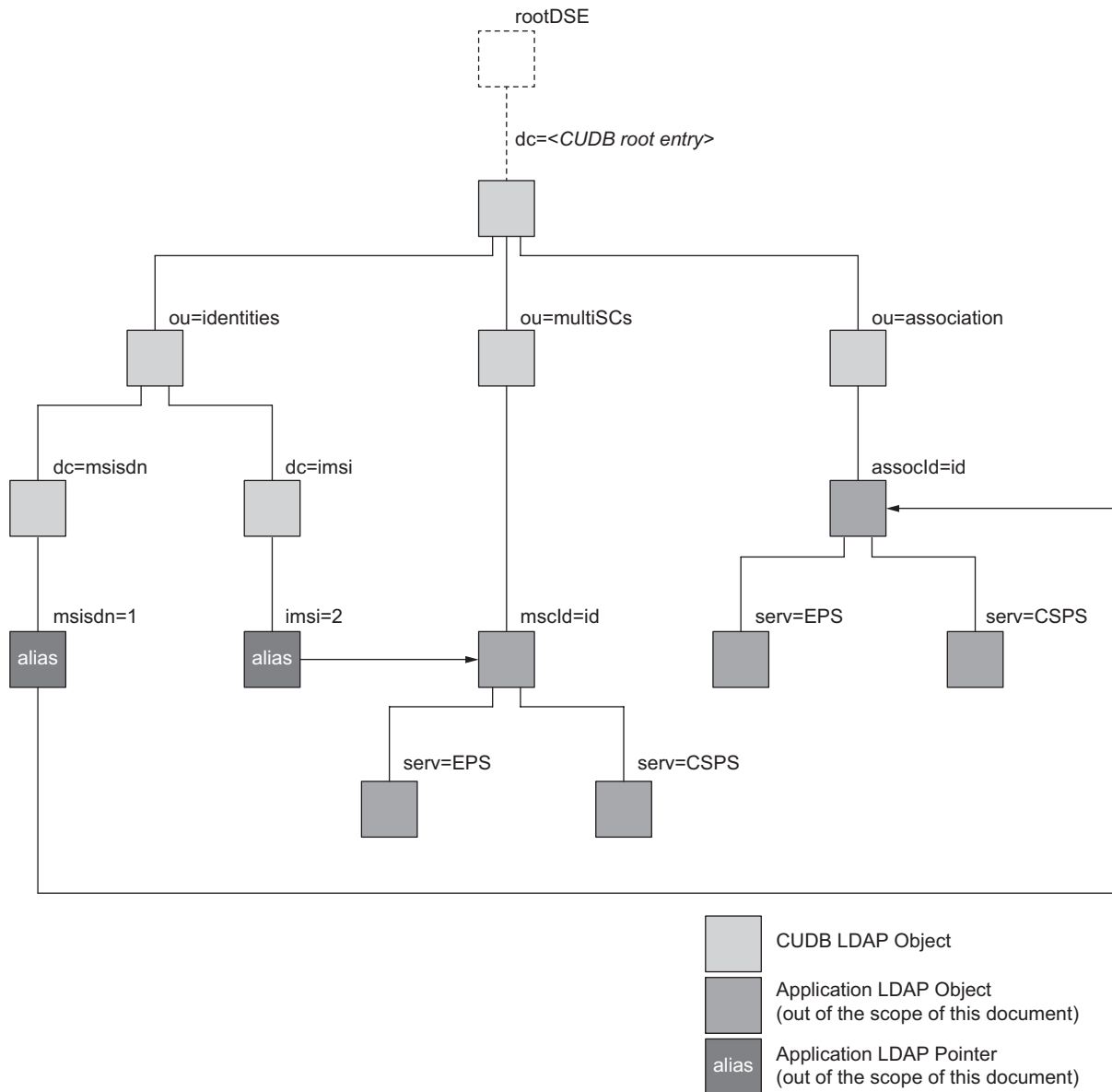
## 3.6 Multi Service Consumer and Association Data

Specific data for each Multi Service Consumer (MSC) may be placed under different entries identified by the `rdn: "mscId = <msc identifier>"`. Different entries under this level are created for the different services the MSC is being serviced by.

Association data specific entries are identified by the `rdn: "assocId=<assoc identifier>"`. Similarly for the MSC's, different entries under this level are created for the different services the MSC is being serviced by.

Figure 4 provides an example of the MSC and association data.





**Figure 4 Multi Service Consumer and Association**

The following entries are defined in CUDB, related to MSC data:

- Msc container, "ou = multiSCs, <CUDB root entry>"
- Msc's, "mscId = <mscId>, ou=multiSCs, <CUDB root entry>"
- MSC data for different services can be placed under mscId entry in "serv=<service>, mscId = <mscId>, ou=multiSCs, <CUDB root entry>", where <service> can take any value. For example, CSPS, Auth and so on.

The following entries are defined in CUDB, related to association data:



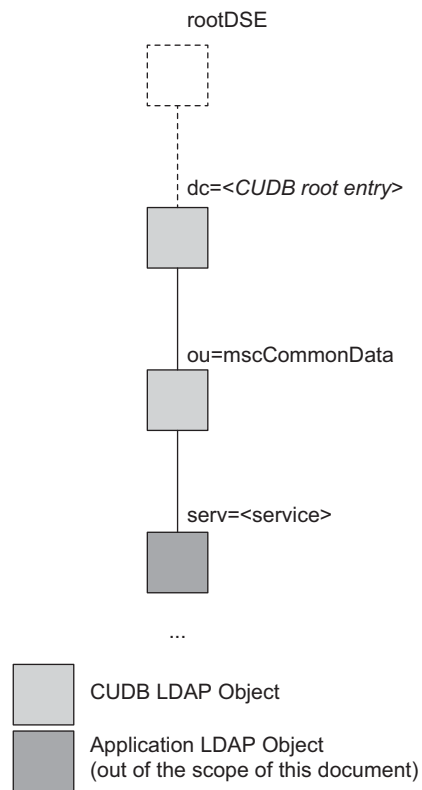
- Association container, "ou = associations, <CUDBroot entry>"
- Associations, "assocId = <assocId>, ou=associations, <CUDB root entry>"
- Association data for different services can be placed under assocId entry in "serv=<service>, assocId = <assocID>, ou=associations, <CUDB root entry>>", where <service> can take any value. For example, CSPA, EPS and so on.

Container entries "ou = multiSCs, <CUDB root entry>" and "ou = associations, <CUDBroot entry>" cannot be deleted since they are parent entries of CUDB Distribution Entries. For more information, refer to the "DEs" section of *CUDB LDAP Data Access*, Reference [1].

## 3.7 MSC Common Data

MSC common data contains data that is shared by a group of MSC's. It is placed under container level entry ou = mscCommonData. Different entries under this level are created for the different services which the MSC is being serviced by. Each of these entries is referenced by all the MSC's which share this common data using LDAP alias.

Figure 5 shows an example of MSC common data.



**Figure 5** *MSC Common Data*

The following entries are defined in CUDB:

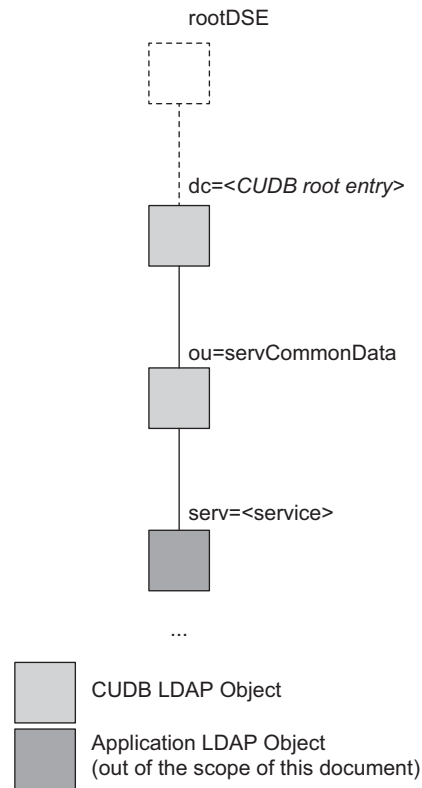
- Msc common data, " ou=mscCommonData, <CUDBroot entry>"

MSC common data for different services is placed under mscCommonData entry in specific msc common data placed under "serv=<service>,ou=mscCommonData, <CUDB root entry>"

## 3.8 Service Common Data

Service common data contains data that is shared at application FE level. Different entries under this level are created for different services.

Figure 6 shows an example of service common data.



**Figure 6** Service Common Data

The following entries are defined in CUDB:

- Service common data, " ou=servCommonData, <CUDB root entry>"

MSC common data for different services can be placed under mscCommonData entry.

- Specific service common data is placed under "serv=<service>,ou=servCommonData, <CUDB root entry>"

## 3.9 LDAP Views

**Note:** The LDAP Data Views function can only be used if the Application Facilitator Value Package is available.

CUDB supports different LDAP data views in order to support flexible data organization. If an LDAP user has an LDAP view assigned to it, then the user will access the data through that view.

Each LDAP data view has its own LDAP schema loaded into the CUDB, building a custom DIT. It also has a mapping file, which describes how the



virtual entries of the specific view are composed from the attributes contained by the core DIT. The CUDB LDAP Data Views function makes it possible to show the data in different DITs accessible through the LDAP interface.

For more information, refer to *CUDB LDAP Data Views*, Reference [2].

### 3.9.1 Concepts

The following list contains the concepts used for LDAP views:

- Core DIT: Current way in CUDB of seeing the data stored in the MySQL cluster through the LDAP interface.
- Real entry: Entry in the CUDB core DIT.
- LDAP Data View (View): Configurable alternative tree-like representation of the data stored in the MySQL cluster, by defining entries composed by mapping of attributes from the core DIT.
- Virtual entry: Entry in a view, whose attributes are mapped from one or several real entries.
- One-to-one mapped entries: Fully writable virtual entries that have a direct matching entry in the core DIT. They have the same content as their core DIT counterparts.





## 4 Attributes and Object Classes

The following sections describe attributes and object classes used in the CUDB.

### 4.1 Attributes

Table 1 lists the attributes defined by CUDB, both for internal use and the benefit of applications storing data inside CUDB.

*Table 1 Attributes*

Attribute Name	Description	Attribute Object Identity	Value Range	Example
assocId	It is a directory string type, single-value attribute that identifies an association.	1.3.6.1.4.1.193.169.2.51	1 - 32 characters	"123456@msim.cudb"



Table 1 Attributes

Attribute Name	Description	Attribute Object Identity	Value Range	Example
CDC	<p>CDC is an LDAP attribute defined in the base CUDB schema (cudb.schema file) which has a customized semantic and especial handling in CUDB for supporting the UDC application FE's Collection Detection Counter (CDC) mechanism.<sup>(1)</sup></p> <p>Client Applications may have this attribute in LDAP entries to avoid race conditions when several clients access LDAP entries at the same time, such as performing a write operation upon an entry previously read when another LDAP client that changed that very same entry between the read and the write operation by the first client. If both clients use CDC as described below, the first client can assert that no other client touched the entry after it read that entry.</p> <p>There are two alternative ways to implement collisions detection using the CDC attribute within an entry:</p> <p><b>Option 1)</b></p> <ul style="list-style-type: none"> <li>Client A reads the LDAP entry, along with its CDC value</li> <li>Client A modifies the LDAP entry and the CDC attribute in the following way: <ul style="list-style-type: none"> <li>first add several new values (add attribute operation) to the multivalued CDC attribute, with values ranging from <code>&lt;CDC_read_value&gt;+1</code> to <code>&lt;CDC_read_value&gt;+N</code>, being N a tolerance factor to detect up to N potential changes made by other clients in the time since the read was done.</li> </ul> </li> </ul> <p>LDAP modify operation</p> <pre>[add/modify/replace/delete operations on attributes belonging to the entry]</pre> <pre>add: CDC: &lt;CDCValueRead&gt;+1 add: CDC: &lt;CDCValueRead&gt;+2 ... add: CDC: &lt;CDCValueRead&gt;+N (as many as concurrent modifications may occur on the entry) - In the same LDAP request, use a modify replace operation on the CDC attribute with &lt;CDC_read_value&gt;+1 replace: CDC: &lt;CDCValueRead&gt;+1 - Perform any other add/modify/replace/de lete operations in the same LDAP request on attributes in that LDAP entry</pre> <ul style="list-style-type: none"> <li>This LDAP operation will fail if one or more other LDAP clients (up to N) changed the LDAP entry (and the CDC value) between the read and the write operations by client A. The way the LDAP modify operation is evaluated makes the 'add' values of CDC fail if CDC already includes any of the values to add (LDAP <code>errorCode=20 attributeOrValueExists</code>). If 1 to N other LDAP clients changed the LDAP entry between the read and the write operations by client A, CDC would also change to a value equal to one of the values in the add operations (<code>&lt;CDC_read_value&gt;+1</code> to <code>&lt;CDC_read_value&gt;+N</code>) by client A, and the write operation will not succeed</li> <li>If the 'add' check passes, the CDC replace operation in the same request will successfully write (one single value) the <code>&lt;CDC_read_value&gt;+1</code> into the CDC, and the rest of the add/modify/replace operations on the entry will go through.</li> </ul> <p><b>Option 2)</b></p> <ul style="list-style-type: none"> <li>Client A reads the entry, along with its CDC value</li> </ul>	1.3.6.1.4.1.193.169.2.102	0-65535	"23"





Table 1 Attributes

Attribute Name	Description	Attribute Object Identity	Value Range	Example
CounterValue	Identifies the value retrieved from the Application Counters database.	1.3.6.1.4.1.193.169.2.401	1 - 32 characters	"4201337"
CUDBNode	Identifies the CUDB node (needed for LDAP proxy support).	1.3.6.1.4.1.193.169.2.101	Any number identifying a CUDB node. For more information about CUDB node configuration, refer to <i>CUDB Node Configuration Data Model Description</i> , Reference [3].	"1"
DSUnitGroup <sup>(2)</sup>	<p>It is an IA5 string type, single-value attribute that identifies the DS UnitGroup where the data related to the distribution entry is to be allocated.</p> <p>In case, the attribute is not included in the add operation, the system would automatically include it with the right value. It should not be modified after the addition of the entry. It has precedence over the ZoneId attribute in case both are included. It has also precedence over a possible distribution algorithm included in a dynamic library overriding the default distribution algorithm. This attribute can be modified when a reallocation procedure is performed. For more information, refer to <i>CUDB Multiple Geographical Areas</i>, Reference [6].</p>	1.3.6.1.4.1.193.169.2.100	<p>Any number identifying a DS Unit Group (DSG) in CUDB.</p> <p>If the DSG is specified when provisioning, it must be a natural number, with no zeros on the left side.</p> <p>For more information on configuration of DSG, refer to <i>CUDB Node Configuration Data Model Description</i>, Reference [3].</p>	"1"
ei	It is a directory string type, single-value attribute that identifies an extensible entry.	1.3.6.1.4.1.193.169.2.300	1 - 32 characters	"GPRS"
mscId	It is a directory string type, single-value attribute that identifies the MSC.	1.3.6.1.4.1.193.169.2.52	1 - 32 characters.	"123456"
serv	It is an IA5 string type, single-value attribute that identifies the service.	1.3.6.1.4.1.193.169.2.2	1 - 32 characters	"CSPS"
ZoneId <sup>(2)</sup>	<p>It is an integer type, single-value attribute that identifies the zone the MSC belongs to.</p> <p>It should be used with care in search filters to retrieve MSCs stored in a given zone because if the attribute is not included in the add operation of the MSC or if it is modified after the add operation, the results of the query could be misleading. This attribute can be modified when a reallocation procedure is performed. For more information, refer to <i>CUDB Multiple Geographical Areas</i>, Reference [6].</p>	1.3.6.1.4.1.193.169.2.105	0-65535	"1"

(1) That CDC mechanism to work properly requires that all LDAP clients accessing the entry with CDC attribute use the logic described above. CDC attribute is defined in the schema as a multi-value integer type in order to support providing several values in the LDAP write requests, although one only value is actually written and persistent in the Database. In practical terms it is stored as single-defined but handled as multivalued while processing LDAP requests, therefore it should be defined as multivalued.

(2) This attribute must be defined in any of the object classes included in the distribution entry.



## 4.2 Object Classes

An object class identifies a set of attributes in an entry in the DIT. Table 2 shows the object classes used in CUDB.

For more information on syntaxes and matching rules, refer to [RFC 4517 LDAP: Syntaxes and Matching Rules](#), Reference [10].

*Table 2 Object Classes*

Object Class Name	Description	Object Class Identity	Required Attributes	
ApplicationCounter	This structural object class is meant to store the application counter value and name retrieved from the database.	1.3.6.1.4.1.193.169.1.202	cn CounterValue	Mandatory Mandatory
CounterGroup	This structural object class is meant to store the name of an application counter group configured in CUDB.	1.3.6.1.4.1.193.169.1.201	cn	Mandatory
CUDBAssociation	It is a structural object class that contains the attributes to handle an association and is included in the ou="associations" container entry.	1.3.6.1.4.1.193.169.1.12	assocId	Mandatory
			ZoneId	Optional
			DSUnitGroup	Optional
CUDBCollisionDetection	It is an auxiliary object class that contains an attribute (CDC) to handle collisions when different LDAP clients update the same entry and might be included in any container entry requiring this type of control.	1.3.6.1.4.1.193.169.1.7	CDC	Optional
CUDBdcObject	It is a structural object class that contains the dc attribute used by identities parent entries.	1.3.6.1.4.1.193.169.1.4	dc	Mandatory
CUDBExtensibleObject	It is an auxiliary object class that is used to handle addition of new entries to the current one.	1.3.6.1.4.1.193.169.1.3	ei	Optional
CUDBMultiserviceConsumer	It is a structural object class that contains the attributes to handle an MSC and is included in the ou="multiSCs" container entry.	1.3.6.1.4.1.193.169.1.11	mscId	Mandatory
			ZoneId	Optional
			DSUnitGroup	Optional
CUDBService	It is a structural object class that contains an attribute to identify a given service and may be used in entries that contain other object classes all of which are auxiliary.	1.3.6.1.4.1.193.169.1.5	serv	Mandatory
CUDBServiceAuxiliary	It is an auxiliary object class that contains an attribute to identify a given service and may be used in entries that contain an structural object class other than this one.	1.3.6.1.4.1.193.169.1.6	serv	Mandatory

## 4.3 Operational Attributes

The CUDB LDAP server supports the following operational attributes, which are detailed in Table 3:

- structuralObjectClass.



- `createTimestamp`.
- `modifyTimestamp`.

For more information about these attributes, refer to [RFC 4512 LDAP: Directory Information Models](#), Reference [11].

**Table 3 Operational Attributes**

Attribute Name	Description	Attribute Object Identity	Value Range	Value Example
<code>structuralObjectClass</code>	This attribute indicates the structural object class of the entry.	1.3.6.1.4.1.1466.115.121.1.38	Any string identifying an object class	<code>structuralObjectClass: lmsImpu</code>
<code>createTimestamp</code>	This attribute stores the creation time of the entry in UTC format. <sup>(1)</sup>	1.3.6.1.4.1.1466.115.121.1.24	Any timestamp in GeneralizedTime format	<code>createTimestamp: 20140412050335Z</code>
<code>modifyTimestamp</code>	This attribute stores the time of the last modification of the entry in UTC format. <sup>(1)</sup> If the entry was never modified, the timestamp will be equal to the value of the <code>createTimestamp</code> attribute.	1.3.6.1.4.1.1466.115.121.1.12	Any timestamp in GeneralizedTime format	<code>modifyTimestamp: 20140329090437Z</code>

(1) These attributes may not be returned for all entries, check *CUDB LDAP Data Access*, Reference [1] for more information.

There are also CUDB-specific operational attributes, which are the following:

- `nodeID`: the ID of the node from where the entry was returned (relevant in case of proxied request).
- `entryDS`: the DS number from which the entry was read.

## 4.4 Limitations

Following are the limitations on attributes and object classes used in CUDB:

- The maximum number of object classes per entry that may be defined is 100.
- Inheritance is not supported between object classes.
- An object class must not have more than 121 attributes (the real limit is 127 but this margin is left in case, future releases require it for internal use).
- Object class capacity is limited to 8 KB. This limit does not include possible blobs in the object class.
- The attributes cannot store more than 4 KB except for octet string attributes which can hold up to 4 GB.



**Note:** The default maximum size for `octetString` and `string` attributes in CUDB is 255 octets. All attributes that are bigger than that or significantly smaller may indicate the minimum upper bound for the size between curly brackets “{ }” after the syntax OID. For more information, refer to [RFC 4512 LDAP: Directory Information Models](#), Reference [11].

`DirectoryString` and `DN` attributes use 3 B per character.

- Distribution entries cannot contain multi-valued attributes.
- The depth of the DIT, that is, the number of sublevels including the root is 12.
- The Relative Distinguish Names (RDNs) of any DN have a limit of 80 unicode characters.
- The maximum length of any DN is 512 unicode characters.
- Alias de-referencing in "subtree" or "onelevel" searches are not supported for the case where the aliases are in subentries under the base DN.
- Aliases in entries under distribution entries pointing to other entries under distribution entries are supported only if both are allocated under the same DSG. For more information, refer to *CUDB LDAP Data Access*, Reference [1].



## 5 CUDB DIT Entries

The following sections describe the predefined entries of the CUDB DIT created at installation time. It also contains entries expected to be created under these predefined entries using LDAP schema.

### 5.1 CUDB Admin Entry

CUDB admin container entry holds data for different CUDB LDAP users. Table 4 shows the CUDB admin entry.

Table 4 CUDB Admin Entry

Entry Name	Object Class Name	Attributes		
		Name	Value Type	Value Example
ou=admin, <CUDB root entry>	<ul style="list-style-type: none"> <li>• top</li> <li>• organizational Unit</li> </ul>	ou	Fixed	"admin"

### 5.2 Identities Container Entry

Identities container entry holds the different identities for the MSC grouped by identity domains. Table 5 shows the identities container entry.

Table 5 Identities Container Entry

Entry Name	Object Class Name	Attributes		
		Name	Value Type	Value Example
ou= identities, <CUDB root entry>	<ul style="list-style-type: none"> <li>• top</li> <li>• organizational Unit</li> </ul>	ou	Fixed	"identities"

### 5.3 Multi Service Consumer Container Entry

Multi service consumer container entry holds the entries containing data for all the MSCs in CUDB. Table 6 shows the multi service consumer container entry.



Table 6 Multi Service Consumer Container Entry

Entry Name	Object Class Name	Attributes		
		Name	Value Type	Value Example
ou= multiSCs, <CUDB root entry>	<ul style="list-style-type: none"><li>• top</li><li>• organizational Unit</li></ul>	ou	Fixed	"multiSCs"

### 5.3.1 Multi Service Consumer Data Entry

Multi service consumer data container entry contains an MSC. Different sub-entries under this level are created for the different services which the MSC is being serviced by. Table 7 shows multi service consumer data entry.

Table 7 Multi Service Consumer Data Entry

Entry Name	Object Class Name	Attributes		
		Name	Value Type	Value Example
mscId= <mscId>, ou=multiSCs, <CUDB root entry>	<ul style="list-style-type: none"><li>• top</li><li>• CUDBMultiServiceConsumer</li></ul>	mscId	Variable	"123456"
		zoneId	Variable	1
		DSUnitGroup	Variable	1

### 5.3.2 Multi Service Consumer Data Service Entry

Multi service consumer data service entry is the container for the service specific related multi service consumer data in CUDB. Table 8 shows multi service consumer data service entry.

Table 8 Multi Service Consumer Data Service Entry

Entry Name	Object Class Name	Attributes		
		Name	Value Type	Value Example
serv=<service>, mscId = <mscId>, ou=multiSCs, <CUDB root entry>	<ul style="list-style-type: none"><li>• top</li><li>• CUDBService</li></ul>	serv	Variable	"CSPS"

## 5.4 Associations Container Entry

Associations container entry is the container for the entries containing data for all the associations in CUDB. Table 9 shows associations container entry.



Table 9 Associations Container Entry

Entry Name	Object Class Name	Attributes		
		Name	Value Type	Value Example
ou= associations, <CUDB root entry>	<ul style="list-style-type: none"> <li>• top</li> <li>• organizational Unit</li> </ul>	ou	Fixed	"associations"

### 5.4.1 Association Data Entry

Association data entry contains an association. Different sub-entries under this level are created for the different services which the association is being serviced by. Table 10 shows association data entry.

Table 10 Association Data Entry

Entry Name	Object Class Name	Attributes		
		Name	Value Type	Value Example
assocId= <assocId>, ou=associations, <CUDB root entry>	<ul style="list-style-type: none"> <li>• top</li> <li>• CUDBAssociation</li> </ul>	assocId	Variable	"123456@msim.cudb"
		ZoneId	Variable	"1"
		DsUnitGroup	Variable	"1"

### 5.4.2 Association Data Service Entry

Association data service entry is the container for the service specific related association data in CUDB. Table 11 shows association data service entry.

Table 11 Association Data Service Entry

Entry Name	Object Class Name	Attributes		
		Name	Value Type	Value Example
ou= associations, <CUDB root entry>	<ul style="list-style-type: none"> <li>• top</li> <li>• CUDBService</li> </ul>	serv	Variable	"CSPS"

## 5.5 Multi Service Consumer Common Data Container Entry

Multi service consumer common data container entry is the container for the MSC common data in CUDB. Table 12 shows multi service consumer common data container entry.



Table 12 Multi Service Consumer Common Data Container Entry

Entry Name	Object Class Name	Attributes		
		Name	Value Type	Value Example
<code>ou=mscCommonData,&lt;CUDB root entry&gt;</code>	<ul style="list-style-type: none"><li>• top</li><li>• organizational Unit</li></ul>	<code>ou</code>	Fixed	"mscCommonData"

### 5.5.1 Multi Service Consumer Common Data Service Entry

Multi service consumer common data service entry is the container for the service specific related service common data in CUDB. Table 13 shows multi service consumer common data service entry.

Table 13 Multi Service Consumer Common Data Service Entry

Entry Name	Object Class Name	Attributes		
		Name	Value Type	Value Example
<code>serv=&lt;service&gt;,ou=mscCommonData,&lt;CUDB root entry&gt;</code>	<ul style="list-style-type: none"><li>• top</li><li>• CUDBService/CUDBServiceAuxiliary</li></ul>	<code>serv</code>	Variable	"CSPS"

## 5.6 Service Common Data Container Entry

Service common data container entry is the container for the service common data in CUDB. Table 14 shows service common data container entry.

Table 14 Service Common Data Container Entry

Entry Name	Object Class Name	Attributes		
		Name	Value Type	Value Example
<code>serv=&lt;service&gt;,ou=servCommonData,&lt;CUDB root entry&gt;</code>	<ul style="list-style-type: none"><li>• top</li><li>• organizational Unit</li></ul>	<code>ou</code>	Fixed	"servCommonData"

### 5.6.1 Service Common Data Service Entry

Service common data service entry is the container for the service specific related service common data in CUDB. Table 15 shows service common data service entry.



*Table 15 Service Common Data Service Entry*

Entry Name	Object Class Name	Attributes		
		Name	Value Type	Value Example
ou=servCommonData,<CUDB root entry>	<ul style="list-style-type: none"><li>• top</li><li>• CUDBService/ CUDBServiceAuxiliary</li></ul>	serv	Variable	"CSPS"





## 6 Operations

The following LDAP operations are used to handle CUDB data:

- Bind.
- Add.
- Delete.
- Modify.
- Search.
- Unbind.

The LDAP Data Views function uses the following LDAP operations when accessing data using the view:

- Modify-add: add attribute to a virtual entry, resulting a new attribute in the existing real entry.
- Modify-delete: delete an optional attribute from a virtual entry, resulting a deleted existing attribute in the real entry.
- Modify-replace: replace an attribute in a virtual entry, resulting a replaced attribute in the existing real entry. (If the attribute does not exist, the request is handled as modify-add. If no value is provided, the request is handled as modify-delete.)
- Search: searches involving data from both PLDB and DSG are not supported.

When accessing data through LDAP data views, special kinds of entries can be used, which are called one-to-one mapped entries.

One-to-one mapped entries are real entries with exactly the same contents as the virtual entries and they can be created, deleted, and their contents modified through views without any restriction other than the ones of a real entry.

**Note:** The LDAP Data Views function can only be used if the Application Facilitator Value Package is available.

There are certain special operations provided by CUDB with respect to standard LDAP operations.

CUDB does not support including the abstract `ObjectClass top` in LDAP modify operations. The abstract `objectClass top` in modify operations is optional, according to LDAP RFCs. In such case, CUDB returns the following error:



```
err=80, msgText=Object class not configured in  
OBJECT_CLASSES table
```

Alias de-referencing in LDAP modify operations for primary identities is a non standard behavior offered by CUDB. This behavior implies that the aliases found when resolving the target DN specified in an LDAP modify operation are always de-referenced if the original DN of the object to be modified contains a primary identity.

Following are the examples of alias de-referencing with a DN that contains a primary identity:

- `serv=csp,IMSI=123456789,dc=imsi,ou=identities, <suffix>`
- `serv=csp,MSISDN=346162681,dc=msisdn,ou=identities,  
<suffix>`

A CUDB LDAP modify over those two entries will result in the modification of the two entries being referenced by **IMSI=123456789** and **MSISDN=346162681** aliases. It means that both, **mscld=id** and **assocld=id** are going to be modified through their corresponding aliases, as depicted in Figure 7:

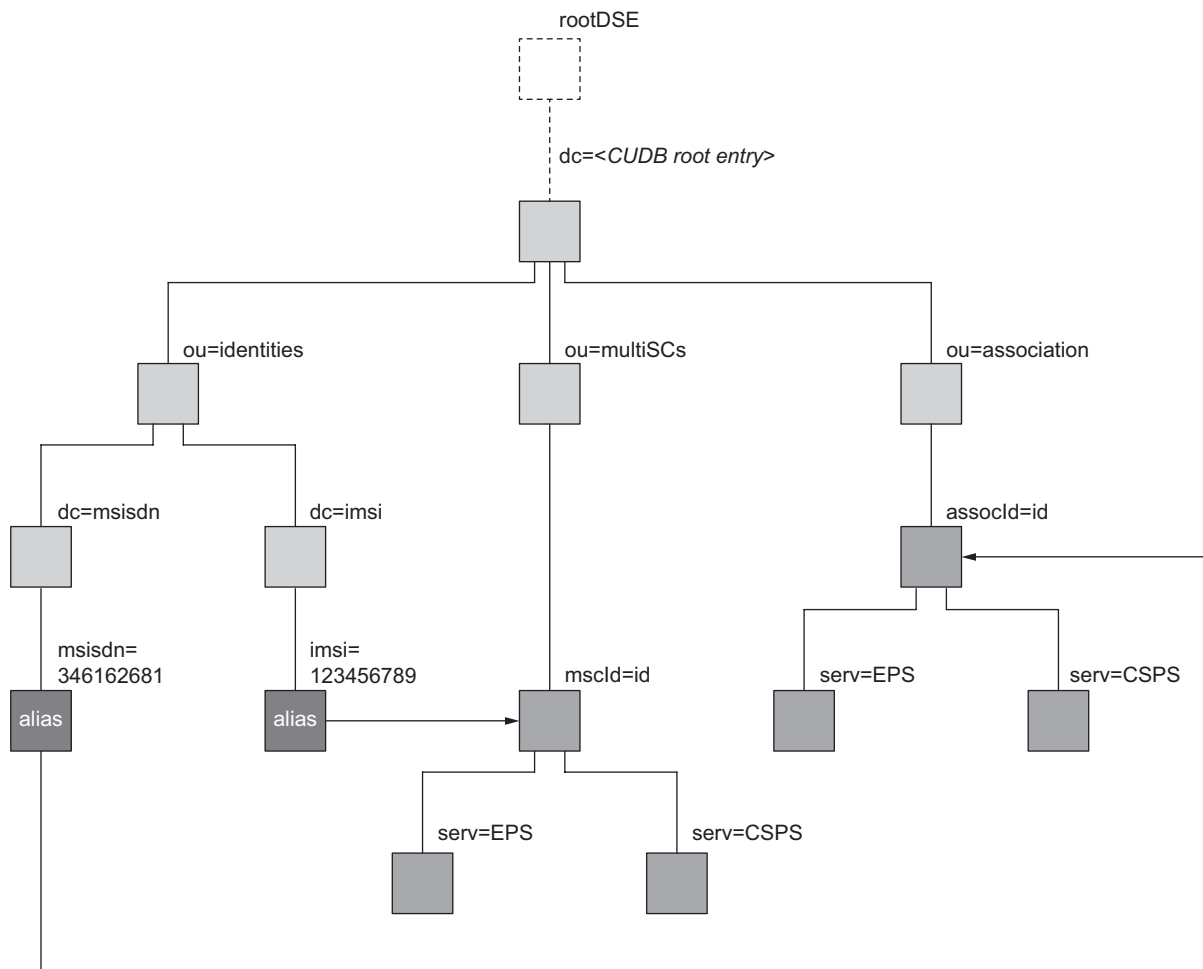


Figure 7 LDAP Modify Operation for Primary Identities

A standard behavior of the LDAP modify would not de-reference, meaning that, in the example above, no modification would take place in **mscld=id** and **assocld=id**. In this case the LDAP modify query returns an error saying that the required entry does not exist.

## 6.1 CUDB Shortcut in Search for Authentication Data

When a search operation is received from any CUDB LDAP client with the baseObject: `dn: serv=Auth, IMSI =<IMSI number>, dc = imsi, ou = identities, <CUDB root entry>` and alias dereferencing is equal to “finding” or “always”, then the operation is interpreted as if baseObject: `dn: IMSI =<IMSI number>, serv=Auth, IMSI =<IMSI number>, dc = imsi, ou = identities, <CUDB root entry>` was received instead.

The same applies for LDAP modify operations for the above baseObject.



## 6.2 Performing LDAP Search Below the Distributed Entry

The CUDB subscriber-centric data model (depicted in Figure 1) enables data consolidation by means of storing applications service profiles right below a distribution entry (`mscId` or `asscoId`).

Client applications only retrieve the data of their own service profiles (their application data model below `serv=<service>` entries) and are unaware of other populated profiles created at the same level below the distribution entry.

The CUDB search engine is optimized for the best performance to serve LDAP search operations that target individual entries, by using the distinguished name of the entry as search base object. Use this type of searches for the best performance.

To enhance the performance of LDAP subtree searches, the CUDB search engine implements an index optimization function for those search operations that drill the LDAP DIT down starting from a start base entry and fetch different LDAP entries. For more information, refer to the “LDAP Massive Search Indexes” section in *CUDB Application Integration Guide*, Reference [8].

To take full advantage of the index search optimization, always set the best possible search base for the LDAP operation, for example at the location in the directory tree from which the LDAP subtree search begins.

As a general recommendation, always start the subtree searches of applications from a `serv=<service>` entry or any other entry below it as the search base object. In other words, do not use subtree searches starting from the distribution entry, otherwise the scanning of every other service profiles and their child entries can cause significant performance drop.

## 6.3 Error Codes

CUDB LDAP interface supports the existing LDAP result codes described in the LDAP protocol specifications. For more information, refer to [RFC 4511 LDAP: The Protocol](#), Reference [12].

Table 16 shows the LDAP result codes in CUDB, describing the possible causes for each error and possible actions about how to handle these errors from the client application perspective.

**Note:** The "Description" column is written in order to provide the most common reasons why the corresponding error is received. Note, that this description does not give all the possible problems. The message diagnostic text can provide more specific information about the error.

For additional information on causes due to overload, mastership change, unavailability or unreachability, refer to the following Facility Descriptions:

- *CUDB LDAP Data Access*, Reference [1]



- *CUDB High Availability*, Reference [7]

Table 16 CUDB LDAP Error Codes

Error Code	Description	Code	Possible Actions
<b>3</b>	Time limit exceeded	<p>In addition to the time limit set by the client exceeded, CUDB adds the following cause:</p> <ul style="list-style-type: none"> <li>• Proxy timeout (search) .</li> </ul>	<ul style="list-style-type: none"> <li>• If the problem is due to proxy timeout, the LDAP client can retry the request.</li> </ul>
<b>11</b>	Administrative limit exceeded	<p>In addition to LDAP causes, CUDB also adds the following causes due to administrative limits (size limit and time limit):</p> <ul style="list-style-type: none"> <li>• Maximum number of DS threads that can access to a local DSG replica.</li> <li>• Proxy timeout (add, delete, modify).</li> </ul>	<ul style="list-style-type: none"> <li>• If the problem is due to proxy timeout, the LDAP client can retry the request.</li> <li>• If the problem is due to maximum number of threads, slow down application/network/terminal retries for those operations rejected from CUDB with the error 80, to stabilize the system faster.</li> </ul>



Table 16 CUDB LDAP Error Codes

Error Code	Description	Code	Possible Actions
51	<ul style="list-style-type: none"><li>• Busy</li><li>• Overload or congestion in the processing layer at CUDB node level.</li><li>• Administrative limit exceeded.</li></ul>	<ul style="list-style-type: none"><li>• The LDAP FE or the PLDB in a specific node are receiving high traffic load above their engineered capacity. CUDB node processing layer components implement overload protection mechanisms and reject incoming operations with this error code if they are unable to process them.</li><li>• The maximum number of threads that can access the local PLDB database in an LDAP FE is reached.</li><li>• The probable cause of reaching the maximum number of threads of the PLDB is an overload in the PLDB.</li><li>• There is one unique DSG deployed in a CUDB system and it is overloaded. When the master replica of the unique DSG in the CUDB system is overloaded, the LDAP FE on the same node returns error code 51 directly.</li></ul>	<ul style="list-style-type: none"><li>• The LDAP client can regulate the traffic sent towards the CUDB node until the congestion situation is over (for example, until no more errors than 51 are received).</li><li>• The LDAP client can retry the request (for example in case of provisioning-related requests).</li></ul>





Table 16 CUDB LDAP Error Codes

Error Code	Description	Code	Possible Actions
52	<ul style="list-style-type: none"><li>• Unavailable</li><li>• The CUDB node is unavailable to process requests.</li></ul>	<p>The CUDB node has not enough available resources, or a failure occurred in a critical component of the node. The error can be caused by one of the below scenarios:</p> <ul style="list-style-type: none"><li>• Minority scenarios, where no master replica for any DSG is reachable.</li><li>• Symmetrical Split scenarios, for search operations performed by provisioning users in the side not hosting the master PLDB replica.</li></ul>	<ul style="list-style-type: none"><li>• Reconnect to another CUDB node, that is, execute a failover towards another CUDB node.</li></ul>



Table 16 CUDB LDAP Error Codes

Error Code	Description	Code	Possible Actions
53	Unwilling to perform.	<ul style="list-style-type: none"><li>• Queries intended for partition (both PL or DS) where the master is unreachable, or there is no replica available.</li><li>• Single Level and wholeSubtree scopes are not supported in this part of the LDAP tree.</li><li>• Operation is not supported within namingContext.</li></ul>	<ul style="list-style-type: none"><li>• Do not perform immediate and frequent reattempts when this error appears. Try to slow down application, network, or terminal retries as much as possible.</li></ul>
80	<ul style="list-style-type: none"><li>• Other error.</li><li>• Problem in some internal component or resource that prevents it to successfully process the request. Other parts of the system may be operating under normal conditions and return successful responses.</li></ul>	<ul style="list-style-type: none"><li>• Data Store (DS) Unit overloaded.</li><li>• Remote CUDB node overloaded or not responding.</li><li>• Subscriber blocked due to reallocation (temporary error).</li><li>• Ongoing master change (temporary error).</li><li>• Application counters backend is busy.</li><li>• LDAP FE is restarting (temporary error).</li></ul>	<ul style="list-style-type: none"><li>• The possible cause of the error is the temporal unavailability of an internal component, or a temporary internal overload of a specific partition or node. To stabilize the system faster, slow down the application, network, or terminal retries for the operations rejected by CUDB with error 80.</li></ul>

For more information on error codes, refer to *CUDB LDAP Data Access*, Reference [1].

## 6.4 LDAP Controls

The LDAPv3 allows clients and servers to use controls as a mechanism for extending an LDAP operation. A control is a way to specify additional

information as part of a request and a response. For more information, refer to [RFC 4511 LDAP: The Protocol](#), Reference [12].

## 6.4.1 ReadMode Control

CUIDB LDAP interface supports ReadMode control with Object ID=1.3.6.1.4.1.193.169.3.3, which is used to override the LDAP read mode user settings for PLDB and DSG.

The ReadMode LDAP control is not supported for LDAP write operations, such as modify, add, or delete. If the Extension Control criticality in the write request is set to `FALSE`, the control is simply ignored and operation is processed normally. If the Extension Criticality is set to `TRUE`, the LDAP request will receive an `err=12, msgText=critical extension is unavailable` message.

In case of distributed or massive searches, that is, the search requests that are distributed to various DSGs, the ReadMode control content is always ignored regardless of criticality.

When ReadMode control is used by applications, the recommended setting for criticality is `FALSE`.

For more information, refer to [RFC 4511 LDAP: The Protocol](#), Reference [12].

The ReadMode control has a numeric value, indicating a combination of `readModeInPL` and `readModeInDS` that will be applied to particular LDAP search request.

*Table 17 Supported ReadMode LDAP Control Values*

Value	For PL	For DS
0	Master always	Master always
1	Local preferred	Master preferred

When a search request is received with ReadMode control, the read mode control values extracted take precedence over `readModeInDS` and `readModeInPL` values configured in LDAP user.





## Glossary

For the terms, definitions, acronyms and abbreviations used in this document, refer to *CUDB Glossary of Terms and Acronyms*, Reference [9].





## Reference List

### **CUDB Documents**

- [1] *CUDB LDAP Data Access*
- [2] *CUDB LDAP Data Views*
- [3] *CUDB Node Configuration Data Model Description*
- [4] *CUDB Application Counters*
- [5] *CUDB System Administrator Guide*
- [6] *CUDB Multiple Geographical Areas*
- [7] *CUDB High Availability*
- [8] *CUDB Application Integration Guide*
- [9] *CUDB Glossary of Terms and Acronyms*

### **Other Documents and Online References**

- [10] *Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules, RFC 4517* <http://www.rfc-editor.org/rfc/rfc4517.txt>
- [11] *Lightweight Directory Access Protocol (LDAP): Directory Information Models, RFC 4512* <http://www.rfc-editor.org/rfc/rfc4512.txt>
- [12] *Lightweight Directory Access Protocol (LDAP): The Protocol, RFC 4511* <http://www.rfc-editor.org/rfc/rfc4511.txt>