

CUDB Node Configuration Data Model Description

CONFIGURATION MODEL

Copyright

© Ericsson AB 2016-2018. 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

1	Introduction	1
1.1	Scope	1
1.2	Revision Information	1
1.3	Target Groups	5
1.4	Typographic Conventions	5
2	CUDB Configuration Model Description	7
2.1	Data Types	7
2.2	Class Hierarchy	8
2.3	System Object Classes	10
2.3.1	Class CudbSystem	10
2.3.2	Class CudbDsGroup	12
2.3.3	Class CudbAppService	13
2.3.4	Class CudbPIGroup	15
2.3.5	Class CudbSystemSecurity	16
2.3.6	Class CudbAutomaticMasterChange	17
2.3.7	Class CudbDsGroupRepairAndResync	17
2.3.8	Class CudbExternalAuthMgmt	18
2.3.9	Class CudbExternalAuthServer	18
2.3.10	Class CudbExternalLogMgmt	19
2.3.11	Class CudbExternalLogServer	20
2.4	Node Object Classes	20
2.4.1	Class CudbLocalNode	21
2.4.2	Class CudbLocalPI	23
2.4.3	Class CudbLocalDs	24
2.4.4	Class CudbProvisioningGatewayConfig	26
2.4.5	Class CudbSecurityMgmt	27
2.4.6	Class CudbLdapCertificates	28
2.4.7	Class CudbSoapCertificates	29
2.4.8	Class CudbRemoteNode	29
2.4.9	Class CudbRemotePI	31
2.4.10	Class CudbRemoteDs	32
2.4.11	Class CudbLogCertificates	34
2.4.12	Class CudbTrafficControlManager	34
2.4.13	Class CudbTrafficBlockingRule	35
2.5	Node Object Structures	36
2.5.1	Structure CudbAsyncActionProgress	36
2.6	LDAP Access Object Classes	37
2.6.1	Class CudbLdapAccess	37
2.6.2	Class CudbLdapUsersMgmt	40
2.6.3	Class CudbLdapUserGroup	40



2.6.4	Class CudbLdapUser	40
2.6.5	Class CudbLdapViewsMgmt	44
2.6.6	Class CudbLdapView	45
2.7	Notifications Object Classes	45
2.7.1	Class CudbNotifications	46
2.7.2	Class CudbNotificationEvent	46
2.7.3	Class CudbNotificationEndPoint	47
2.7.4	Class CudbNotificationObjectClass	48
2.7.5	Class CudbNotificationAttr	49
2.8	PG Object Classes	50
2.8.1	Class CudbProvisioningGatewayMgmt	50
2.8.2	Class CudbProvGatewayEndPoint	51
2.9	CUDB Administrative Operations	52
2.9.1	applyConfig	53
2.9.2	updateUserInfo	56
2.9.3	cancelApplyConfig	59
3	Initial Configuration	61
3.1	Considerations	61
4	Configuration Modification Procedure	63
4.1	Preconditions	63
4.2	Object Model Modification Procedure	63
4.2.1	Modification Procedure Using CUDB Configuration CLI	63
4.2.2	Modification Procedure Using NETCONF	67
	Glossary	73
	Reference List	75



1 Introduction

This document describes the configuration data model of the Ericsson Centralized User Database (CUDb) and the configuration modification procedures. This document applies to all CUDb nodes in the CUDb system, as all CUDb nodes are configured in the same way.

1.1 Scope

This document covers the following topics:

- Configuration model description.
- Initial configuration.
- Configuration modification procedure.

1.2 Revision Information

Rev. A

This document is based on 1/19202-CSH 109 067/9 with the following changes:

- Replaced parent class information with full path of configuration objects throughout the document.
- Section 2 on page 7: Updated description of Read only attributes in Table 1.
- Section 2.1 on page 7: Updated Table 2 with information on `EcimStruct`, `EcimAction`, and `EcimEnumeration` data types.
- Section 2.2 on page 8: Added new classes `CudbTrafficControlManager` and `CudbTrafficBlockingRule` in Figure 1.
- Section 2.3.3 on page 13: Added a Warning.
- Section 2.4.1 on page 20: Updated Table 14 with information on `applyConfigStatus` attribute.
- Section 2.4.1 on page 20, Section 2.4.2 on page 23, Section 2.4.3 on page 24, Section 2.4.8 on page 29, and Section 2.4.9 on page 31: Added new data type values to the corresponding attributes in tables.
- Section 2.4.2 on page 23: Updated `CudbLocalPl` class description.
- Section 2.4.3 on page 24: Updated `CudbLocalDs` class description and data type in Table 16.



- Section 2.4.6 on page 28 and Section 2.4.7 on page 29: Updated sections with reference to *CUDB Security and Privacy Management*, Reference [12].
- Section 2.4.9 on page 31: Updated `CudbRemotePl` class description.
- Section 2.4.10 on page 32: Updated data type in Table 23.
- Section 2.4.12 on page 34: Added section on the `CudbTrafficControlManager` class.
- Section 2.4.13 on page 35: Added section on the `CudbTrafficBlockingRule` class.
- Section 2.5 on page 36: Added section on Ericsson Common Information Model (ECIM) structures.
- Section 2.6.1 on page 37: Added a constraint for `ldapRootPassword` attribute in Table 28.
- Section 2.6.1 on page 37 and Section 2.6.4 on page 40: Updated note with information on new `applyConfig` action.
- Section 2.6.4 on page 40, Section 2.6.5 on page 44, and Section 2.6.6 on page 45: Updated information on function availability.
- Section 2.9 on page 52: Added section on administrative operations.
- Section 4.2.1 on page 63 and Section 4.2.2 on page 66: Added new subsections to Section 4.2 on page 63.

Rev. B

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

- Section 2.4.1 on page 20: Added new attribute to Table 14: `cudbCounterPublishingPeriod` and updated VIP names in Attribute Name column.
- Section 2.6.4 on page 40: Updated the Local Preferred (LP) value in Table 31. Added a value combination for the `readModeInPL` and `readModeInDS` LDAP user attributes and a note about the configuration of `readModeInDS=LP`.
- Section 2.7.2 on page 46, Section 2.7.3 on page 47, Section 2.7.4 on page 48, and Section 2.7.5 on page 49: Updated class information.
- Section 2.7.4 on page 48: Updated `dn` attribute information in Table 37.
- Section 4.2.1 on page 63 and Section 4.2.2 on page 66: Added information on deleting parent classes.



Rev. C

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

- Structural and content rearrangement throughout the document.
- Section 2.3.5 on page 16: Updated the description of `lockoutPeriod` attribute in Table 7.
- Section 2.5.1 on page 36: Updated the description of `timeOfLastStatusUpdate` attribute in Table 27.
- Section 2.6.4 on page 40: Updated attribute name information for `readModeInDS` and `readModeInPL` with LDAP `ReadMode` control information in Table 31.
- Section 2.7.4 on page 48: Updated `dn` attribute information in Table 37 with support for extended POSIX regular expressions.

Rev. D

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

- Section 2.3.1 on page 10: Added new attribute, `localReadsDSReplicationSafetyThreshold`, to Table 3.
- Section 2.3.9 on page 18: Updated `tlsEnabled` attribute and added `tlsMode` attribute to Table 11.
- Section 2.4.1 on page 20: Updated `networkElementName` attribute in Table 14.

Rev. E

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

- Section 2.3.1 on page 10: Updated the description of the `localReadsDSReplicationDelayThreshold` attribute in Table 3.
- Section 2.6.4 on page 40: Added new attribute, `localReadsDsReplicationDelayThreshold`, to Table 31. Updated data type of `cudbLdapViewId` attribute with new `ldapViewName` attribute in Table 31.
- Section 2.6.6 on page 45: Updated data type of `cudbLdapViewId` attribute in Table 33 and added new `ldapViewName` attribute. Added note about `cudbLdapViewId` attribute.

Rev. F

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

- Section 2.1 on page 7: Added new data type, `EcimPassphraseString`



- Section 2.6.1 on page 37: Updated data type of `ldapRootPassword` attribute. Removed note.
- Section 2.9.1.5 on page 55, Section 2.9.2.5 on page 58: Updated reference to point to *CUDB System Administrator Guide*.
- Section 4.2.2 on page 66: Updated Step 1 and Step 2.
- Section 4.2.2.1 on page 68: Removed reference.

Rev. G

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

- Section 2.1 on page 7: Updated Data type and Description of `CudbIPAddress` in Table 2.
- Section 2.3.9 on page 18: Updated Data type of `primaryServer` and `secondaryServer` in Table 11.
- Section 2.3.11 on page 20: Updated Data type of `externalLogServerIp` in Table 13.
- Section 2.4.1 on page 20: Updated Data Type of `cudbLocalNodeId`, `cudbVIP`, `oamVIP` and `trafficVIP` in Table 14.
- Section 2.4.8 on page 29: Updated Data type of `cudbVIP`, `oamVIP` and `trafficVIP` in Table 21.
- Section 2.6.1 on page 37: Updated LDAP root user password storage form of `nodeLdapAuth` attribute in Table 28.
- Section 4.2.2 on page 66: Updated `applyConfig` status command.

Rev. H

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

- Section 2.4.13 on page 35: Updated Data type of `blockedVIP` in Table 26.

Rev. J

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

- Section 2.1 on page 7: Updated Table 2.
- Section 2.3.1 on page 10: Added `customerId`, `deploymentId`, and `productNumber` attributes to Table 3, and updated the description and data type of `replicationTimeDelayAlarmThreshold` attribute.
- Section 2.3.9 on page 18: In Table 11, updated Data type in the rows for `primaryServer` and `secondaryServer`.



- Section 2.3.11 on page 20: In Table 13, updated Data type in the row for `externalLogServerIp`.
- Section 2.4.1 on page 20: In Table 14, updated Data type in the rows for `cudbVIP`, `oamVIP` and `trafficVIP`.
- Section 2.4.4 on page 26: In Table 17, updated Data type in the row for `pgNodeIpAddresses`.
- Section 2.4.8 on page 29: In Table 21, updated Data type in the rows for `cudbVIP`, `oamVIP` and `trafficVIP`.
- Section 2.4.13 on page 35: In Table 26, updated Data type in the row for `blockedVIP`.
- Section 2.6.1 on page 37: Updated `ldapAttrIndexes` attribute, and removed Tablenote 1 and Tablenote 2 in Table 28.
- Section 2.7.3 on page 47: In Table 36, updated Data type in the third row for `URI`.
- Section 2.8.2 on page 51: In Table 40, updated Data type in the row for `replayRequestURL` and `replayStatusURL`.

1.3 Target Groups

This document is intended for network operators to configure the CUDB system. It is also intended for Ericsson installation engineers to configure the initial CUDB system.

1.4 Typographic Conventions

Typographic conventions can be found in the following document:

- *Typographic Conventions*





2 CUDB Configuration Model Description

This section describes the data objects and attributes configured within the CUDB system. The objects form a hierarchy, wherein each object is identified by a unique Relative Distinguished Name (RDN). Creating or deleting objects, or changing object attributes requires knowledge of object's RDN.

The whole configuration model is created from the `ManagedElement` root class which is a part of the Common Operation and Maintenance (COM) model. For more information on COM, refer to *COM Management Guide*, Reference [24].

Table 1 shows the table structure used to describe data objects and attributes in this document.

Table 1 Table Structure

Attribute Name	Data Type	Properties
Attribute name followed by its description.	<ul style="list-style-type: none"> Syntax shows the format of the attribute. Range shows the possible values of the attribute. Constraints are the restrictions applied to the attribute value. Default value/example. <p>For more information on data types, see Section 2.1 on page 7.</p>	<ul style="list-style-type: none"> Optional: The attribute can be set optionally. Mandatory: The attribute must be present at object instance creation. Restricted: The attribute can only be set at object instance creation. Read only: The attributes can only be read. The value of the attribute is obtained from the system, read from the BC server. If the connection to the BC server is down, the value is set as <code>emptySet</code>, returning empty square brackets: <code>[]</code>. Read/Write: The attribute can be read and written. Write only: The attribute can only be written. Multivalued: The attribute contains multiple values following the format <code>[<value>, <value2>...]</code>. These values can be defined sequentially, especially when adding a new one.

Note: The `userLabel` attribute is an optional, `EcimString` type attribute that can be used freely by operators. The `userLabel` attribute is omitted from the tables describing object attributes.

2.1 Data Types

Table 2 shows data types used in defining attributes in the CUDB configuration model.



Table 2 Data Types

Data Type	Description
IpDNSAddress	<p>Specifies v4 or v6 IP address in corresponding notation. IPv4 and IPv6 addresses are both supported in CUDB, but their combination is not supported. CUDB system can be either IPv4 or IPv6 based.</p> <p>Example: 130.100.92.154 or 2001:cdba:0000:0000:0000:0000:3257:9652</p> <p><i>RFC 791</i>, Reference [25] describes the syntax details for v4 and <i>RFC 4291</i>, Reference [26] describes the syntax details for v6.</p>
EcimAction	Administrative operation.
EcimBoolean	<p>Possible values:or</p> <ul style="list-style-type: none">• true• false
EcimEnumeration	A sequence of literals, containing a name and a value for each item.
EcimPassphraseString	A sequence of characters representing a password. When configured through CLI, it must be entered twice. The value entered is not echoed. It appears masked in log.
EcimPasswordString	A sequence of characters representing a password. It appears masked in log.
EcimString	A sequence of characters.
EcimStruct	A group containing multiple data types that are handled within the same scope.
EcimUint32	Unsigned 32-bit integer.
NumericMaxInclusive2	EcimUint32 with minimum value equal to zero and maximum value equal to 2.
NumericMaxInclusive100	EcimUint32 with minimum value equal to zero and maximum value equal to 100.
NumericMaxInclusive255	EcimUint32 with minimum value equal to zero and maximum value equal to 255.
NumericMaxInclusive65535	EcimUint32 with minimum value equal to zero and maximum value equal to 65535.
NumericMinInclusive1	EcimUint32 with minimum value equal to 1.
NumericRangeInclusive1to3600000	Specifies a time period between 1ms - 3600000 ms (1 hour).
NumericString	A string that represents a number.
Time	Represents the local time as portrayed by the international standard ISO 8601.

2.2 Class Hierarchy

This section shows the CUDB class hierarchy. Figure 1 shows the CUDB classes containing the CUDB information model and the cardinality of each class.





2.3 System Object Classes

This section describes the classes for generic configuration in the CUDB system.

2.3.1 Class CudbSystem

The `CudbSystem` class is the root class of the CUDB Managed Object Model (MOM). There is only one instance per CUDB node, which cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:

`ManagedElement=1,CudbSystem=1`

Table 3 shows the attributes of the `CudbSystem` class.

Table 3 Class *CudbSystem*

Attribute Name	Data Type	Properties
<code>automaticServiceContinuity</code> Attribute for enabling or disabling the automatic triggering of Service Continuity for minority scenarios. Refer to the <i>Service Continuity for Asymmetrical Split Scenarios</i> section of <i>CUDB High Availability</i> , Reference [2] for more information.	<code>EcimBoolean</code> Default value: <code>false</code>	Optional Read/Write
<code>backboneReliability</code> Deprecated: has no functional behavior.	<code>EcimBoolean</code> Default value: <code>true</code>	Optional Restricted
<code>binlogExpireDays</code> Binlog files older than the value set in this attribute are eligible to be purged by an age-based purge.	<code>NumericMaxInclusive255</code> Range: 0–255 Default value: 2	Optional Read/Write
<code>cudbSystemId</code> Identifies the instance of this class.	<code>EcimString</code> Range: 1 Value: 1	Mandatory Restricted
<code>customerId</code> Uniquely identifies the customer. The same identifier that was used for obtaining the CUDB software for the customer must be used here.	<code>EcimString</code> Example: 942544	Mandatory Restricted
<code>defaultZone</code> Includes all CUDB nodes that are not explicitly included in any other geographical zone. For more information, refer to <i>CUDB Multiple Geographical Areas</i> , Reference [1].	<code>EcimUInt32</code> Example: 1	Optional Read/Write
<code>deploymentId</code> Uniquely identifies the deployed customer system. The hardware resource name must be used when activating the licenses for the CUDB system from the Supply organization.	<code>EcimString</code> Example: CUDB_TiLab_CUFTL101	Mandatory Restricted



Table 3 Class CudbSystem

Attribute Name	Data Type	Properties
dsClusterDropRatioAlarmThreshold Defines a threshold for the Data Store (DS) cluster, that is the number of operations dropped due to overload in the DS cluster over the total number of operations intended to be processed by the DS cluster. If the DS cluster drop ratio goes above this threshold, the Storage Engine, High Load in DS alarm is raised. For more information, refer to <i>Storage Engine, High Load In DS</i> , Reference [7].	NumericMaxInclusive100 Range: 0–100 Default value: 5	Optional Read/Write
ldapFrontEndDropRatioAlarmThreshold Defines a threshold for the Lightweight Directory Access Protocol (LDAP) Front End (FE), that is the number of operations dropped due to overload in the LDAP FEs over the total number of operations received in the node. If the LDAP FE drop ratio goes above this threshold, the LDAP Front End, High Load in LDAP Processing Layer alarm is raised. For more information, refer to <i>LDAP Front End, High Load in LDAP Processing Layer</i> , Reference [14].	NumericMaxInclusive100 Range: 0–100 Default value: 5	Optional Read/Write
localReadsDSReplicationDelayThreshold Defines the maximum threshold (in seconds) for replication delay value, which is used to determine if the slave replica is too far behind the master replica. If the slave replica is too far behind, the data will not be read locally. This applies only to users whose DS read mode is set to Local Preferred (LP). This threshold is overridden by the user attribute localReadsDsReplicationDelayThreshold, if it is available. See section Section 2.6.4 on page 40 for more information.	EcimUInt32 Range: 10–7200 Default value: 300	Optional Read/Write
mimName The name of the model.	EcimString Default value: cudb	Optional Restricted
mimRelease The release of the model. Not used.	EcimString Default value: 0	Optional Restricted
mimVersion The version of the model. Not used.	EcimString Default value: 1	Optional Restricted
pldbDropRatioAlarmThreshold Defines a threshold for the Processing Layer Database (PLDB), that is, the number of operations dropped due to overload in the PLDB over the total number of operations intended to be processed by the PLDB. If the PLDB drop ratio goes above this threshold, the Storage Engine, High Load in PLDB alarm is raised. For more information, refer to <i>Storage Engine, High Load in PLDB</i> , Reference [8].	NumericMaxInclusive100 Range: 0–100 Default value: 5	Optional Read/Write
productNumber The CXP product number for the CUDB software, without any suffix or versioning information.	EcimString Example: CXP9020214	Mandatory Restricted
provisioningAssurance Attribute for enabling or disabling “Provisioning assurance after CUDB mastership change” feature at any time.	EcimBoolean Constraint: At least one instance of CudbProvGatewayEndPoint class must exist before activating the feature. Example: true	Mandatory Read/Write

**Table 3** Class *CudbSystem*

Attribute Name	Data Type	Properties
reallocationBlockSize Sets the granularity for the number of DEs to be reallocated in a reallocation operation. The number of reallocated DEs is a multiple of this attribute. This attribute has some effect only in case the feature <i>CUDB Subscription Reallocation</i> , Reference [3] is active.	EcimUint32 Range: 1–10000 Example: 500	Mandatory Read/Write
replicationTimeDelayAlarmThreshold Defines a threshold in seconds for the replication delay between a slave PLDB or DS cluster slave replica and the master PLDB or DS cluster master replica it is replicating from, expressed as the estimated time needed for the slave replica to catch up with the master replica. By default Replication Delay Monitoring is disabled (attribute is set to 0) and no alarms are raised regardless of the value of the replication delay. To activate monitoring, a different value, other than the default one, must be set for the threshold. That value must be tuned to the particular system, since the delay is impacted by network overload, master or slave replica server overload, rotating the binlog that can temporarily cause a rise in delay, and so on, or conditions, which can vary from system to system. This value must be dimensioned by taking into account the delays in the network and the delays in the CUDB processing. It must have a value high enough to avoid intermittent alarms. If the replication delay goes above this threshold, the <i>Storage Engine, Replication Delay Too High In PLDB</i> or <i>Storage Engine, Replication Delay Too High In DS</i> alarm is raised. For more information, refer to <i>Storage Engine, Replication Delay Too High In PLDB</i> , Reference [6] and <i>Storage Engine, Replication Delay Too High In DS</i> , Reference [5].	EcimUint32 Default value: 0 Example: 15	Optional Read/Write

2.3.2 Class *CudbDsGroup*

Several DSs are logically grouped in a CUDB system into a DS Unit Group (DSG). The *CudbDsGroup* class represents a DSG and it contains the configuration parameters common to all DSs belonging to the same DSG. There are as many instances of this class as DS groups defined in a CUDB system, which can be deleted from the configuration model.

The full path to the instances of this class is as follows:

ManagedElement=1, CudbSystem=1, CudbDsGroup=<CUDB_Ds_Group_Id>

Table 4 shows the attributes of the *CudbDsGroup* class.

Table 4 Class *CudbDsGroup*

Attribute Name	Data Type	Properties
accessPort Sets the port for internally SQL cluster data access. The suggested value for this port follows this rule: 15000 + (DS Group number * 10). Do not modify this parameter once the node is providing traffic service.	NumericMaxInclusive65535 Range: 0–65535 Constraint: It must be unique in the configuration data model. Example: for DS Group 8, 15080 = 15000 + 8 * 10	Mandatory Restricted



Table 4 Class CudbDsGroup

Attribute Name	Data Type	Properties
cudbDsGroupId Identifies the instance of this class.	EcimString Range: 1–255 Example: 1	Mandatory Restricted
masterReplicationChannel1Port Sets the port where the master DS of the DSG listens for replication purposes. This port corresponds to replication channel 1. The suggested value for this port follows this rule: $15000 + (\text{DS Group number} * 10) + 1$. Do not modify this parameter once the node is providing traffic service.	NumericMaxInclusive65535 Range: 0–65535 Constraint: It must be unique in the configuration data model. Example: for DS Group 8, $15081 = 15000 + 8 * 10 + 1$	Mandatory Restricted
masterReplicationChannel2Port Sets the listening port on the master DS of the DSG for replication purposes. This port corresponds to replication channel 2. The suggested value for this port follows this rule: $15000 + (\text{DS Group number} * 10) + 2$. Do not modify this parameter once the node is providing traffic service.	NumericMaxInclusive65535 Range: 0–65535 Constraint: It must be unique in the configuration data model. Example: for DS Group 8, $15082 = 15000 + 8 * 10 + 2$	Mandatory Restricted
memoryEligibleThreshold Sets the percentage of occupation under which the DSG is selectable for accepting distributed data from other DSGs while reallocating.	NumericMaxInclusive100 Range: 0–100 Constraint: It must be lower than memoryWarningThreshold. Example: 25	Mandatory Read/Write
memoryWarningThreshold Sets the percentage of occupation over which the reallocation of distributed data is recommended.	NumericMaxInclusive100 Range: 0–100 Constraint: It must be greater than memoryEligibleThreshold. Example: 75	Mandatory Read/Write

2.3.3 Class CudbAppService

The `CudbAppService` class represents a single service or application FE, providing a set of object classes and attributes. There are as many instances of this class as application FEs using the CUDB system, which cannot be deleted from the configuration model.



Warning!

When creating new `CudbAppService` objects, ensure that the order in which the new objects are created follows the increasing order of their `cudbAppServiceId` attributes. This order must be the same for every node in the system.

The full path to the instances of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbAppService=<CUDB_App_Service_Id>
```

Table 5 shows the attributes of the `CudbAppService` class.

Table 5 Class *CudbAppService*

Attribute Name	Data Type	Properties
<code>appSrvName</code> Name of the application FE. It also prefixes all object classes present in the schema that belongs to this application FE, unless the specific application FE requires that object classes present in the schema do not have prefixes.	<code>EcimString</code> Example: <code>csps</code>	Mandatory Restricted
<code>cudbAppServiceId</code> Identifies the instance of this class. The value for this identifier is relevant as it defines the order in which LDAP schemas are loaded in the system. Services or application FEs having schemas that have dependencies with other application FEs or services schemas have to contain an identifier with a higher value than the ones on which they depend. It is also important that the schema set in the <code>CudbLdapAccess</code> instance is loaded before any service or application FE schema.	<code>EcimString</code> Example: <code>1</code>	Mandatory Restricted
<code>ldapAppSrvSchema</code> File name (without file path) where the LDAP schema for this application FE is stored. For more information on LDAP schema, refer to <i>CUDB LDAP Interwork Description</i> , Reference [10]. This attribute is modified whenever a schema update is performed. For more information about schema updates, refer to <i>CUDB Application Schema Update</i> , Reference [11].	<code>EcimString</code> Example: <code>csps.schema</code>	Mandatory Read/Write
<code>sqlAppSrvDsSchema</code> File name (without file path) where the internal data format for this application FE is stored in the DS. This attribute is modified whenever a schema update is performed. For more information about schema updates, refer to <i>CUDB Application Schema Update</i> , Reference [11].	<code>EcimString</code> Example: <code>csps-ds.sql</code>	Mandatory Read/Write
<code>sqlAppSrvPlSchema</code> File name (without file path) where the internal data format for this application FE is stored in the PLDB. This attribute is modified whenever a schema update is performed. For more information about schema updates, refer to <i>CUDB Application Schema Update</i> , Reference [11].	<code>EcimString</code> Example: <code>csps-pl.sql</code>	Mandatory Read/Write



2.3.4 Class CudbPlGroup

All processing layers (PLs) are logically grouped in the CUDB system into a PL group. The `CudbPlGroup` class represents a single PL group. It contains the configuration parameters common to all PLs that belong to the PL group. There is only one instance of this class for all the PLs in the CUDB system, which cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:

ManagedElement=1, CudbSystem=1, CudbPlGroup=1

Table 6 shows the attributes of the `CudbPlGroup` class.

Table 6 Class *CudbPlGroup*

Attribute Name	Data Type	Properties
accessPort Sets the port for internally accessing the cluster data through SQL. The suggested value for this port is 15000. Do not modify this parameter once the node is providing traffic service.	NumericMaxInclusive65535 Range: 0–65535 Constraint: It must be unique in the configuration data model. Example: 15000	Mandatory Restricted
cudbPlGroupId Identifies the instance of this class.	EcimString Range: 1 Example: 1	Mandatory Restricted
masterReplicationChannel1Port Sets the listening port on the master PL of the PL group for replication purposes. This port corresponds to replication channel 1. The suggested value for this port is 15001. Do not modify this parameter once the node is providing traffic service.	NumericMaxInclusive65535 Range: 0–65535 Constraint: It must be unique in the configuration data model. Example: 15001	Mandatory Restricted
masterReplicationChannel2Port Sets the listening port on the master PL of the PL group for replication purposes. This port corresponds to replication channel 2. The suggested value for this port is 15002. Do not modify this parameter once the node is providing traffic service.	NumericMaxInclusive65535 Range: 0–65535 Constraint: It must be unique in the configuration data model. Example: 15002	Mandatory Restricted
memoryWarningThreshold Sets the percentage of occupation at which the <i>Storage Engine, Memory Usage Too High In PLDB, Warning</i> , Reference [9] alarm is raised.	NumericMaxInclusive100 Range: 0–100 Example: 80	Mandatory Read/Write



2.3.5 Class CudbSystemSecurity

The `CudbSystemSecurity` class contains all security related configuration in CUDB. For more information on security, refer to *CUDB Security and Privacy Management*, Reference [12].

Only one instance of this class is present in each CUDB node, which cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:

ManagedElement=1, CudbSystem=1, CudbSystemSecurity=1

Table 7 shows the attributes of the `CudbSystemSecurity` class.

Table 7 Class *CudbSystemSecurity*

Attribute Name	Data Type	Properties
<code>cudbSystemSecurityId</code> Identifies the instance of this class.	<code>EcimString</code> Range: 1 Example: 1	Mandatory Restricted
<code>lockoutPeriod</code> Specifies how long the account is locked for after the number of unsuccessful login attempts indicated by <code>maxNumFailedLogins</code> attribute, in seconds.	<code>EcimUInt32</code> Default value: 21600	Optional Read/Write
<code>maxNumFailedLogins</code> Number of unsuccessful logins before a user account is blocked.	<code>EcimUInt32</code> Default value: 5	Optional Read/Write
<code>minPasswordLength</code> The minimum Operation and Maintenance (OAM) password length.	<code>EcimUInt32</code> Default value: 8	Optional Read/Write
<code>minPasswordNonRepeat</code> The minimum number of unique passwords before a password can be repeated.	<code>EcimUInt32</code> Default value: 12	Optional Read/Write
<code>secureLdapProxy</code> Indicates that the LDAP client initiates proxy connection using Transport Layer Security (TLS).	<code>EcimBoolean</code> Default value: false	Optional Read/Write
<code>secureMySQLReplication</code> Indicates that the database cluster client initiates connection between nodes using TLS.	<code>EcimBoolean</code> Default value: false	Optional Read/Write
<code>systemMonitorSafeMode</code> Deprecated: Has no functional behavior.	<code>EcimBoolean</code> Default value: true	Optional Read/Write
<code>tlsCaCertificatesFile</code> Full path of the file containing a list of certificates for trusted Certificate Authorities (CAs). Among those, it must include the CAs that signed the certificates stored in CUDB node.	<code>EcimString</code> Default value: ""	Optional Read/Write



2.3.6 Class CudbAutomaticMasterChange

The `CudbAutomaticMasterChange` class contains all configuration related to Automatic Mastership Change (AMC) in CUDB. Only one instance of this class is present in each CUDB node, which it cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:

`ManagedElement=1, CudbSystem=1, CudbAutomaticMasterChange=1`

Table 8 shows the attributes of the `CudbAutomaticMasterChange` class.

Table 8 Class *CudbAutomaticMasterChange*

Attribute Name	Data Type	Properties
cudbAutomaticMasterChangeId Identifies the instance of this class.	EcimString Range: 1 Example: 1	Mandatory Restricted
enabled Specifies if AMC is enabled or not.	EcimBoolean Default value: false	Optional Read/Write
maxReplicationTimeDelay Defines a threshold in milliseconds for the replication delay between a slave PLDB or slave DSG cluster replica and the master PLDB or master DSG cluster replica the slave is replicating from. The threshold is expressed as the estimated time needed for the slave replica to catch up with the master replica. If the replication delay of the preferred master is above the threshold, the preferred master will not automatically take the mastership.	EcimUint32 Default value: 3000	Optional Read/Write
timeWindowStart This attribute, along with <code>timeWindowEnd</code> , is used to define the daily time interval in which the AMC process is allowed to run. In relation to the <code>timeWindowEnd</code> , this attribute can be configured as follows: <ul style="list-style-type: none"> If <code>timeWindowStart = timeWindowEnd</code>, AMC is always allowed to run. If <code>timeWindowStart < timeWindowEnd</code>, AMC can run during the defined time period every day. If <code>timeWindowStart > timeWindowEnd</code>, AMC can run between <code>timeWindowStart</code> and <code>timeWindowEnd</code> on the next day. 	Time Default value: 00:00:00	Optional Read/Write
timeWindowEnd This attribute, along with <code>timeWindowStart</code> , is used to define a daily time interval in which the AMC process is allowed to run. See the <code>timeWindowStart</code> attribute for more information on the relationship between the two attributes.	Time Default value: 00:00:00	Optional Read/Write

2.3.7 Class CudbDsGroupRepairAndResync

The `CudbDsGroupRepairAndResync` class is used to contain configuration parameters related to Selective Replica Check, Data Repair, and Self-Ordered Backup and Restore in CUDB. Only one instance of this class is present in each CUDB node, which cannot be deleted.



The full path to the instance of this class is as follows:

ManagedElement=1,CudbSystem=1,CudbDsGroupRepairAndResync=1

Table 9 shows the attributes of the `CudbDsGroupRepairAndResync` class.

Table 9 Class `CudbDsGroupRepairAndResync`

Attribute Name	Data Type	Properties
<code>automaticBackupRestoreEnabled</code> Specifies whether Self-Ordered Backup and Restore is enabled or not.	<code>EcimBoolean</code> Default value: <code>true</code>	Optional Read/Write
<code>autoSRCCAndDREnabled</code> Specifies whether automatic execution of Selective Replica Check and Data Repair is enabled or not.	<code>EcimBoolean</code> Default value: <code>true</code>	Optional Read/Write
<code>cudbDsGroupRepairAndResyncId</code> Identifies the instance of this class.	<code>EcimString</code> Range: 1 Example: 1	Mandatory Restricted

2.3.8 Class `CudbExternalAuthMgmt`

The `CudbExternalAuthMgmt` class contains the attributes used for the configuration of the CUDB OAM Centralized Authentication System Support function.

The full path to the instance of this class is as follows:

ManagedElement=1,CudbSystem=1,CudbExternalAuthMgmt=1

Table 10 shows the attributes of the `CudbExternalAuthMgmt` class.

Table 10 Class `CudbExternalAuthMgmt`

Attribute Name	Data Type	Properties
<code>CudbExternalAuthMgmt</code> Identifies the instance of this class.	<code>EcimString</code> Range: 1 Example: 1	Mandatory Restricted
<code>enabled</code> Specifies if the feature is activated or not.	<code>EcimBoolean</code> Default value: <code>false</code>	Optional Read/Write

2.3.9 Class `CudbExternalAuthServer`

The `CudbExternalAuthServer` class contains the parameters needed to connect to an external authentication server, as the CUDB OAM Centralized Authentication System Support feature describes.

The full path to the instance of this class is as follows:



ManagedElement=1,CudbSystem=1,CudbExternalAuthMgmt=1,CudbExternalAuthServer=1

Table 11 shows the attributes of the CudbExternalAuthServer class.

Table 11 Class CudbExternalAuthServer

Attribute Name	Data Type	Properties
baseDn DN used for searches.	EcimString	Mandatory Read/Write
bindDn DN user for binding.	Ecimstring	Optional Read/Write
bindPassword Password used for binding.	EcimPasswordString	Optional
cudbExternalAuthServerId Identifies the instance of this class.	EcimString Range: 1 Value: 1	Mandatory Restricted
primaryServer IP of the primary external authentication server.	IpDNSAddress Constraint: Strictly IP address syntax, host names are not allowed. Example: 10.1.5.15	Mandatory Read/Write
secondaryServer IP of the secondary external authentication server.	IpDNSAddress Constraint: IPv4 or IPv6 address syntax, host names are not allowed. Example: 10.1.5.15 or 2001:cdba:0000:0000:0000:0000:3257:9652	Optional Read/Write
tlsEnabled Specifies if TLS will be used in connection with the external server. If TLS is enabled, the certification authority certificate must be part of the file introduced in the tlsCaCertificatesFile attribute of the CudbSystemSecurity class. How the client establishes a secure connection to the external server is defined with the tlsMode attribute.	EcimBoolean Default value: false	Optional Read/Write
tlsMode Indicates if a secure session will start from an insecure session: STARTTLS (port 389), or if it will be started directly from: LDAPS (port 636).	EcimString Allowed values: • STARTTLS • LDAPS Default value: STARTTLS	Optional Read/Write

2.3.10 Class CudbExternalLogMgmt

The CudbExternalLogMgmt class contains the configuration for the Centralized Security Event Logging function.

The full path to the instance of this class is as follows:



ManagedElement=1, CudbSystem=1, CudbExternalLogMgmt=1

Table 12 shows the attributes of the CudbExternalLogMgmt class.

Table 12 Class CudbExternalLogMgmt

Attribute Name	Data Type	Properties
cudbExternalLogMgmtId Identifies the instance of this class.	EcimString Range: 1 Value: 1	Mandatory Restricted
enabled Specifies if the function is activated or not.	EcimBoolean Default value: false	Optional Read/Write

2.3.11 Class CudbExternalLogServer

The CudbExternalLogServer class contains the parameters needed to send security logs to an external server, as the Centralized Security Event Logging function describes.

The full path to the instance of this class is as follows:

ManagedElement=1, CudbSystem=1, CudbExternalLogMgmt=1, CudbExternalLogServer=1

Table 13 shows the attributes of the CudbExternalLogServer class.

Table 13 Class CudbExternalLogServer

Attribute Name	Data Type	Properties
cudbExternalLogServerId Identifies the instance of this class.	EcimString Range: 1 Value: 1	Mandatory Restricted
externalLogServerIp IP address of the external log server.	IpDNSAddress Constraint: IPv4 or IPv6 address syntax, host names are not allowed. Example: 10.1.5.15 or 2001:cdba:0000:0000:0000:0000:3257:9652	Mandatory Read/Write
externalLogServerPort Port used by the external log server.	NumericMaxInclusive65535 Range: 0–65535	Mandatory Read/Write

2.4 Node Object Classes

This section describes the classes for configuration of CUDB nodes in the CUDB system.



2.4.1 Class CudbLocalNode

The `CudbLocalNode` class specifies the CUDB local node configuration. Only one instance of this class is present in each CUDB node, which cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>
```

Table 14 shows the attributes of the `CudbLocalNode` class.

Warning!

Restricted attributes, such as **trafficVIP**, **oamVIP**, or **cudbVIP**, can require a reinstallation in order to be reverted. Pay attention to their initial setting to avoid that situation.

Warning!

The values of **trafficVIP**, **oamVIP**, and **cudbVIP** attributes *must* be different because of the traffic separation and must fulfill the values introduced in the network configuration set in eVIP.

Table 14 Class `CudbLocalNode`

Attribute Name	Data Type	Properties
applyConfigStatus Shows the status of an asynchronous <code>applyConfig</code> administrative operation. Note: The <code>progressPercentage</code> attribute of the structure is never updated. For more information, see Section 2.5.1 on page 36.	EcimStruct Note: The attribute value points to a certain instance of the <code>CudbAsyncActionProgress</code> structure.	Mandatory Restricted
cudbCounterPublishingPeriod Determines how frequently the performance management 3GPP XML output files containing CUDB counters are published. This attribute only affects CUDB counters. It does not affect application counters.	EcimUInt8 Allowed values: <ul style="list-style-type: none"> 5 15 Default value: 15	Optional Read/Write



Table 14 Class CudbLocalNode

Attribute Name	Data Type	Properties
cudbLocalNodeId Identifies the instance of this class.	EcimString Range: Integer, 0 < cudbLocalNodeId < 256 Constraint: It must be different for each CUDB node in a CUDB system. Example: 1	Mandatory Restricted
cudbVIP Specifies the default virtual IP address that other CUDB nodes have to use to exchange any kind of traffic with this local node. It is usually referred to as SITE_VIP.	IpDNSAddress Constraint: IPv4 or IPv6 address syntax, host names are not allowed. Example: 10.1.5.15 or 2001:cdba:0000:0000:0000:0000:3257:9652	Mandatory Restricted
enabled When set to false, the local CUDB node is hidden to other nodes. It neither answers any control messages from other CUDB nodes nor sends any messages. For more information, refer to <i>CUDB High Availability</i> , Reference [2].	EcimBoolean Default value: true	Optional Read/Write
hwType The type of hardware used in the node.	EcimString Allowed values: <ul style="list-style-type: none">EBS_GEP3EBS_GEP5vCUDB_2CPU_6GBvCUDB_16CPU_47GB Example: EBS_GEP5	Mandatory Restricted
networkElementName Network Element Name required for this CUDB node. It is used to form the file name of counter output files (generated by the performance management subsystem). It is unique per node. Network Element Name is part of all blades or Virtual Machines (VMs) prompt in CUDB node.	EcimString Example: CUDB_1 Note: Valid characters are A to Z, a to z, and 0 to 9, using no spaces. Also characters - and _ are allowed. The length is restricted to 2-40 characters.	Mandatory Read/Write
oamVIP Specifies the virtual IP address that any external application must use to exchange OAM related traffic with this local node. It is usually referred to as OAM_VIP.	IpDNSAddress Constraint: IPv4 or IPv6 address syntax, host names are not allowed. Example: 10.1.5.14 or 2001:cdba:0000:0000:0000:0000:3257:9653	Mandatory Restricted
siteId This is the site where the node is located. For more information, refer to <i>CUDB High Availability</i> , Reference [2].	EcimUint32 Example: 1	Mandatory Restricted
systemMonitorKey Deprecated: Has no functional behavior.	EcimString Default value: ""	Optional Read/Write



Table 14 Class CudbLocalNode

Attribute Name	Data Type	Properties
trafficVIP Specifies the virtual IP address that any external application can use to exchange LDAP traffic with this local node. It is usually referred to as FE_VIP.	IpDNSAddress Constraint: IPv4 or IPv6 address syntax, host names are not allowed. Example: 10.1.5.13 or 2001:cdba:0000:0000:0000:0000:3257:9654	Mandatory Restricted
updateUserInfoStatus Shows the status of an asynchronous updateUserInfo administrative operation. Note: The progressPercentage attribute of the structure is never updated. For more information, see Section 2.5.1 on page 36.	EcimString Note: The attribute value points to a certain instance of the CudbAsyncActionProgress structure.	Mandatory Restricted
zone This is the zone to which the node belongs. For more information, refer to <i>CUDB Multiple Geographical Areas, Reference</i> [1].	EcimUint32 Default value: 0	Optional Restricted

2.4.2 Class CudbLocalPl

The CudbLocalPl class represents a PLDB unit in the local CUDB node. This class is optional and only one instance per CudbLocalNode can be present. At least one instance of CudbLocalPl or CudbRemotePl must exist per site.

The full path to the instance of this class is as follows:

```
ManagedElement=1, CudbSystem=1, CudbLocalNode=<CUDB_Local_Node_Id>, CudbLocalPl=1
```

Table 15 shows the attributes of the CudbLocalPl class.

Table 15 Class CudbLocalPl

Attribute Name	Data Type	Properties
cudbLocalPlId Identifies the instance of this class.	EcimString Range: 1 Value: 1	Mandatory Restricted
enabled Determines if the specified local PLDB cluster is taken into account in the CUDB system including where it is hosted for LDAP traffic purposes, AppCounters computing purposes and system data backup procedure. Disabling the local PLDB cluster results in logical disconnection of this local CUDB node from the CUDB system.	EcimBoolean Example: false	Mandatory Read/Write



Table 15 Class CudbLocalPI

Attribute Name	Data Type	Properties
instancePriority Priority assigned to this PLDB instance in the CUDB system PL Group. In the same conditions, this attribute gives the order of preference for each cluster to be elected as master. Top priority is 1.	NumericMinInclusive1 Syntax: Integer higher than 0. Constraint: It must be unique across all PLDB storage instances in the whole CUDB system. Example: 2	Mandatory Restricted
instanceState State of the local PLDB replica. Following are the possible values: <ul style="list-style-type: none"> 0: absent, meaning cluster is down. 1: active and degraded, meaning cluster is working but some of its data nodes are down. 2: active and non-degraded, meaning cluster is working perfectly. 	NumericMaxInclusive2 Range: 0, 1, 2 Example: 1	Read only
isMaster Reports if this PLDB unit is acting as master in the PL Group.	EcimBoolean Example: false	Read only
memoryUsage Amount of memory (%) used in the PL Group.	NumericMaxInclusive100 Example: 58	Read only
numAssignedNodes Number of PLDB dedicated blades or VMs.	EcimUInt32 Range: 4–16 (only even values), when hwType is EBS_GEP3 Range: 2–16 (only even values), when hwType is EBS_GEP5, vCUDb_2CPU_6GB, or vCUDb_16CPU_47GB Example: 4	Mandatory Restricted

2.4.3 Class CudbLocalDs

The `CudbLocalDs` class represents DS units hosted in the CUDB node. There are as many instances of this class as DS units exist in the CUDB local node, which can be deleted from the configuration model.

The full path to the instances of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDb_Local_Node_Id>,CudbLocalDs=<CUDb_Local_Ds_Id>
```

Table 16 shows the attributes of the `CudbLocalDs` class.



Table 16 Class CudbLocalDs

Attribute Name	Data Type	Properties
cudbLocalDsId Identifies the instance of this class. Specifies the DS cluster physical position inside the physical CUDB node where this DS unit is allocated.	EcimString Range: 1–15 when the CudbLocalNode that contains this attribute has CudbLocalPl class created. Range: 1–17 when the CudbLocalNode that contains this attribute does not have CudbLocalPl created. Constraint: It must be defined consecutively starting from 1. Example: 1	Mandatory Restricted
dsGroupId DS Group Identity that this DS instance belongs to. See Section 2.3.2 on page 12.	NumericMinInclusive1 Range: 1–255 Constraint: It must correspond to an existing instance of CudbDsGroup. Example: 3	Mandatory Restricted
enabled Determines if the specified local DS cluster is visible in the CUDB system including where it is hosted for LDAP traffic purposes, AppCounters computing purposes, and system data backup procedure.	EcimBoolean Example: false	Mandatory Read/Write
instancePriority Priority assigned to this storage instance in the CUDB system DS Group. In the same conditions, this attribute gives the order of preference for each cluster to be elected as master. The lower is the positive value, the higher the priority is. Top priority is 1.	NumericMinInclusive1 Syntax: Integer higher than 0. Constraint: It must be unique across all DS storage instances belonging to the same DS Group (with same dsGroupId in the whole CUDB system). Example: 2	Mandatory Restricted
instanceState State of the local DS replica. Following are the possible values: <ul style="list-style-type: none"> 0: absent, meaning cluster is down. 1: active and degraded, meaning cluster is working but some of its data nodes are down. 2: active and non-degraded, meaning cluster is working perfectly. 	EcimUInt32 Range: 0, 1, 2 Example: 1	Read only
isMaster Reports if this DS unit is acting as master for the DS group it belongs to.	EcimBoolean Example: false	Read only
memoryUsage Amount of database memory (%) used in the DS Unit.	NumericMaxInclusive100 Example: 58	Read only



2.4.4 Class CudbProvisioningGatewayConfig

The `CudbProvisioningGatewayConfig` class is used to specify the IP addresses and credentials to connect to a Provisioning Gateway (PG) in the CUDB local node containing it. The connection with the PG notifies backup related events to the PG. Only one instance of this class is present in each CUDB node, which can be deleted from the configuration model.

The full path to the instance of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>,CudbProvisioningGatewayConfig=1
```

Table 17 shows the attributes of the `CudbProvisioningGatewayConfig` class.

Table 17 Class `CudbProvisioningGatewayConfig`

Attribute Name	Data Type	Properties
<code>cudbProvisioningGatewayConfigId</code> Identifies the instance of this class.	<code>EcimString</code> Range: 1 Value: 1	Mandatory Restricted



Table 17 Class *CudbProvisioningGatewayConfig*

Attribute Name	Data Type	Properties
<p><code>pgNodeIpAddresses</code></p> <p>This attribute can have multiple values, each representing the list of OAM_VIP IP addresses and including the notification ports for a specific PG node.</p>	<p>EcimString</p> <p>Syntax: [<code><ip1>[:JMXport1:JNDIport1]</code>; <code><ipn>[:JMXportn:JNDIportn]</code>]</p> <p>where 1...n are the IP addresses for each PG node.</p> <p>Separator for IP addresses within a node: semicolon (;)</p> <p>If IPv6 address is used in combination with ports, it needs to be put in square brackets.</p> <p>As square brackets are standard syntax for defining multivalue attributes, if any attribute contains them, it needs to be put in quotation marks (").</p> <p>Example: One node with two IP addresses: [10.1.33.141:9994:4099;10.1.33.142:8994:8099]</p> <p>Example : Two nodes with two IP addresses each [10.1.33.141:9994:4099;10.1.33.142:8994:8099, 10.1.33.143:9994:4099;10.1.33.144:8994:8099]</p> <p>Example : Two nodes with two IPv6 addresses each ["[2001:1b70:8294:3d84::1]:8994:8099;[2001:1b70:8294:3d84::2]:8994:8099", 2001:1b70:8294:3d84::3;2001:1b70:8294:3d84::4]</p> <p>Note: If JMX and JNDI ports are not defined default values are used: "9994" for JMX and "4099" for JNDI.</p>	<p>Mandatory</p> <p>Read/Write</p> <p>multivalued</p>
<p><code>pgUserName</code></p> <p>The user name of the PG nodes.</p>	<p>EcimString</p> <p>Example: <code>pgUser</code></p>	<p>Mandatory</p> <p>Read/Write</p>
<p><code>pgUserPassword</code></p> <p>The user password of the PG nodes. It is not stored in plain text.</p>	<p>EcimPasswordString</p> <p>Example: <code>0pgUser1Pwd</code></p>	<p>Mandatory</p> <p>Write only</p>

2.4.5 Class *CudbSecurityMgmt*

The *CudbSecurityMgmt* class is used to contain the classes that configure security mechanisms based on TLS. Only one instance of this class is present in each CUDB node, which cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:



ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>,CudbSecurityMgmt=1

Table 18 shows the attributes of the CudbSecurityMgmt class.

Table 18 Class CudbSecurityMgmt

Attribute Name	Data Type	Properties
cudbSecurityMgmtId Identifies the instance of this class.	EcimString Range: 1 Value: 1	Mandatory Restricted

2.4.6 Class CudbLdapCertificates

The CudbLdapCertificates class is used to specify the configuration of the TLS used to secure LDAP communications using LDAPv3. For more information on security, refer to *CUDB Security and Privacy Management, Reference* [12].

Only one instance of this class is present in each CUDB node, which cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:

ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>,CudbSecurityMgmt=1,CudbLdapCertificates=1

Table 19 shows the attributes of the CudbLdapCertificates class.

Table 19 Class CudbLdapCertificates

Attribute Name	Data Type	Properties
cudbLdapCertificatesId Identifies the instance of this class.	EcimString Range: 1 Value: 1	Mandatory Restricted
tlsCertificateFile The path and file name of the file containing LDAP server certificate. Enable TLS with this attribute.	EcimString Default value: "" Example: /cluster/certificates/ldapfe/servercert.pem	Optional Read/Write
tlsCertificateKeyFile The path and file name containing the private key that matches the certificate stored in tlsCertificateKeyFile. Enable TLS with this attribute.	EcimString Default value: "" Example: /cluster/keys/ldapfe/serverkey.pem	Optional Read/Write



2.4.7 Class CudbSoapCertificates

The `CudbSoapCertificates` class contains the configuration of the Hypertext Transfer Protocol Secure (HTTPS/TLS) to secure communications using Simple Object Access Protocol (SOAP) to send notifications. For more information on security, refer to *CUDB Security and Privacy Management*, Reference [12].

Only one instance of this class is present in each CUDB node, which cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:

```
ManagedElement=1, CudbSystem=1, CudbLocalNode=<CUDB_Local_Node_Id>, CudbSecurityMgmt=1, CudbSoapCertificates=1
```

Table 20 shows the attributes of the `CudbSoapCertificates` class.

Table 20 Class *CudbSoapCertificates*

Attribute Name	Data Type	Properties
<code>cudbSoapCertificatesId</code> Identifies the instance of this class.	EcimString Range: 1 Value: 1	Mandatory Restricted
<code>tlsCertificateFile</code> The path and file name of the file containing SOAP client certificate.	EcimString Default value: "" Example: /cluster/certificates/soap/servercert.pem	Optional Read/Write
<code>tlsCertificateKeyFile</code> The path and file name of the file containing the private key that matches the certificate stored in <code>tlsCertificatesFile</code> .	EcimString Default value: "" Example: /cluster/keys/soap/serverkey.pem	Optional Read/Write

2.4.8 Class CudbRemoteNode

The `CudbRemoteNode` class represents the CUDB remote nodes. There are as many instances of this class as the number of CUDB nodes minus one in the CUDB system. Instances of this class can be deleted if the node to delete is not the only one in its site.

The full path to the instances of this class is as follows:

```
ManagedElement=1, CudbSystem=1, CudbRemoteNode=<CUDB_Remote_Node_Id>
```

Table 21 shows the attributes of the `CudbRemoteNode` class.



Warning!

Restricted attributes, such as, `trafficVIP`, `oamVIP` or `cudbVIP` can require a reinstallation in order to be reverted. Pay attention to their initial setting to avoid this situation.

Warning!

By default, the values of **trafficVIP**, **oamVIP**, and **cudbVIP** attributes *must* be equal and must fulfill the values introduced in the network configuration set in `eVIP`. For more information, refer to *CUDB Node Network Description*, Reference [15].

Table 21 Class *CudbRemoteNode*

Attribute Name	Data Type	Properties
<code>cudbRemoteNodeId</code> Identifies the instance of this class. It must be different for each CUDB node in a CUDB system. It is the node identification of the remote CUDB node.	<code>NumericString</code> Range: 1–255. Example: 5	Mandatory Restricted
<code>cudbVIP</code> Specifies the default virtual IP address the local node uses to exchange any kind of traffic with the remote node represented by this class instance.	<code>IpDNSAddress</code> Constraint: IPv4 or IPv6 address syntax, host names are not allowed. Example: 10.1.5.15 or 2001:cdba:0000:0000:0000:0000:3257:9652	Mandatory Restricted
<code>enabled</code> When set to false, it disables the CUDB remote node without deleting this instance object, that is, a non-existing node. For more information, refer to <i>CUDB High Availability</i> , Reference [2].	<code>EcimBoolean</code> Default value: true	Optional Read/Write
<code>hwType</code> The type of hardware used in the node.	<code>EcimString</code> Allowed values: <ul style="list-style-type: none">EBS_GEP3EBS_GEP5vCUDB_2CPU_6GBvCUDB_16CPU_47GB Example: EBS_GEP5	Mandatory Read/Write



Table 21 Class *CudbRemoteNode*

Attribute Name	Data Type	Properties
oamVIP Specifies the virtual IP address the local node uses to exchange OAM related traffic with the remote node represented by this class instance. This virtual IP address must be the same as the one stated in attribute <code>cudbVIP</code> in this <i>CudbRemoteNode</i> configuration class instance. However, it might be different in particular cases or customizations in which OAM related traffic is delivered through a different and separated transport network.	IpDNSAddress Constraint: IPv4 or IPv6 address syntax, host names are not allowed. Example (default configuration): 10.1.5.15 or 2001:cdba:0000:0000:0000:0000:3257:9652	Mandatory Restricted
siteId This is the site where the node is located. For more information, refer to <i>CUDB High Availability</i> , Reference [2].	EcimUint32 Example: 1	Mandatory Restricted
systemMonitorKey Deprecated: Has no functional behavior.	EcimString Default value: ""	Optional Read/Write multivalued
trafficVIP Specifies the virtual IP address the local node uses to exchange LDAP proxy traffic with the remote node represented by this class instance. This virtual IP address must be the same as the one stated in attribute <code>cudbVIP</code> in this <i>CudbRemoteNode</i> configuration class instance. However, it might be different in particular cases in which LDAP proxy traffic between CUDB nodes is delivered through a different and separated transport network.	IpDNSAddress Constraint: IPv4 or IPv6 address syntax, host names are not allowed. Example (default configuration): 10.1.5.15 or 2001:cdba:0000:0000:0000:0000:3257:9652	Mandatory Restricted
zone This is the zone to which the node belongs. For more information, refer to <i>CUDB High Availability</i> , Reference [2].	EcimUint32 Default value: 0	Optional Restricted

2.4.9 Class *CudbRemotePl*

The *CudbRemotePl* class represents PLDB units in remote CUDB nodes. This class is optional and only one instance per *CudbRemoteNode* can be present (and must be created in the same commit than its parent). At least one instance of *CudbLocalPl* or *CudbRemotePl* must exist per site.

The full path to the instances of this class is as follows:

```
ManagedElement=1, CudbSystem=1, CudbRemoteNode=<CUDB_Remote_Node_Id>, CudbRemotePl=1
```

Table 22 shows the attributes of the *CudbRemotePl* class.

Table 22 Class *CudbRemotePl*

Attribute Name	Data Type	Properties
cudbRemotePlId Identifies the instance of this class.	EcimString Range: 1 Value: 1	Mandatory Restricted



Table 22 Class CudbRemotePl

Attribute Name	Data Type	Properties
enabled Determines if this remote PLDB cluster is taken into account in the CUDB system including where it is hosted for LDAP traffic purposes, AppCounters computing purposes and system data backup procedure.	EcimBoolean Example: false	Mandatory Read/Write
instancePriority Priority assigned to this PLDB instance in the CUDB system PL group. Under the same conditions, this attribute gives the order of preference for each cluster to be elected as master. The lower the positive value, the higher the priority is. Top priority is 1.	NumericMinInclusive1 Syntax: Integer higher than 0. Constraint: It must be unique across all PLDB storage instances in the entire CUDB system.	Mandatory Restricted
instanceState State of the remote PLDB replica. Following are the possible values: <ul style="list-style-type: none"> 0: absent, meaning cluster is down. 1: active and degraded, meaning cluster is working but some of its data nodes are down. 2: active and non-degraded, meaning cluster is working perfectly. 	EcimUint32 Range: 0, 1, 2 Example: 1	Read only
isMaster Reports if this PLDB unit is acting as master in the PL Group.	EcimBoolean Example: false	Read only
numAssignedNodes Number of PLDB dedicated blades or VMs.	EcimUint32 Range: 4–16 (only even values), when hwType is EBS_GEP3 Range: 2–16 (only even values), when hwType is EBS_GEP5, vCUDb_2CPU_6GB, or vCUDb_16CPU_47GB Example: 4	Mandatory Read/Write

2.4.10 Class CudbRemoteDs

The `CudbRemoteDs` class represents DS units in remote CUDB nodes. There are as many instances of this class as DS units exist in the CUDB remote node, which can be deleted from the configuration model.

The full path to the instances of this class is as follows:

```
ManagedElement=1, CudbSystem=1, CudbRemoteNode=<CUDB_Remote_Node_Id>, CudbRemoteDs=<CUDB_Remote_Ds_Id>
```

Table 23 shows the attributes of the `CudbRemoteDs` class.



Table 23 Class CudbRemoteDs

Attribute Name	Data Type	Properties
cudbRemoteDsId Identifies the instance of this class. Specifies the DS cluster physical position inside the physical CUDB node where this DS unit is allocated.	EcimString Range: 1–15 when the CudbRemoteNode that contains this attribute has CudbRemotePl class created. Range: 1–17 when the CudbRemoteNode that contains this attribute does not have CudbRemotePl class created. Example: 1 Constraint: It must be defined consecutively starting from 1.	Mandatory Restricted
dsGroupId DS Group Identity (positive and starting from 1) that this DS instance belongs to. See Section 2.3.2 on page 12.	NumericMinInclusive1 Constraint: It must correspond to an existing instance of CudbDsGroup. Example: 3	Mandatory Restricted
enabled Specifies if this remote DS cluster is visible in the local CUDB node for LDAP traffic purposes, AppCounters computing purposes and system data backup procedure.	EcimBoolean Example: false	Mandatory Read/Write
instancePriority Under the same conditions, this attribute gives the order of preference for master election assigned to this storage instance in the CUDB system DS Group The lower the positive value, the higher the priority is. Top priority is 1.	NumericMinInclusive1 Syntax: Integer higher than 0. Constraint: It must be unique across all DS storage instances belonging to the same DS Group (with same dsGroupId in the whole CUDB system). Example: 2	Mandatory Restricted
instanceState State of the remote DS replica. Following are the possible values: <ul style="list-style-type: none"> 0: absent, meaning cluster is down. 1: active and degraded, meaning cluster is working but some of its data nodes are down. 2: active and non-degraded, meaning cluster is working perfectly. 	EcimUInt32 Range: 0, 1, 2 Example: 1	Read only
isMaster Reports if this DS unit is acting as master for the DS group it belongs to.	EcimBoolean Example: false	Read only
memoryUsage Amount of database memory (%) used in the DS Unit.	EcimUInt32 Example: 58	Read only



2.4.11 Class CudbLogCertificates

The `CudbLogCertificates` class contains the TLS configuration used to secure the communication with an external log server, as the Centralized Security Event Logging function describes.

The full path to the instance of this class is as follows:

```
ManagedElement=1, CudbSystem=1, CudbLocalNode=<CUDB_Local_Node_Id>, CudbSecurityMgmt=1, CudbLogCertificates=1
```

Table 24 shows the attributes of the `CudbLogCertificates` class.

Table 24 Class `CudbLogCertificates`

Attribute Name	Data Type	Properties
<code>cudbLogCertificatesId</code> Identifies the instance of this class.	<code>EcimString</code> Range: 1 Value: 1	Mandatory Restricted
<code>tlsCertificateFile</code> The path and name of the file containing the client certificate used for TLS communication by the Centralized Security Event Logging function.	<code>EcimString</code> Example: /cluster/certificates/log/cert.pem	Mandatory Read/Write
<code>tlsCertificateKeyFile</code> The path and name of the file containing the private key that matches the certificate stored in <code>tlsCertificateFile</code> .	<code>EcimString</code> Example: /cluster/keys/log/key.pem	Mandatory Read/Write
<code>logServerName</code> The name of the server, the same name must be used in the server certificate generation.	<code>EcimString</code> Default value: "*" Example: logserver	Optional Read/Write

2.4.12 Class CudbTrafficControlManager

The `CudbTrafficControlManager` class represents a container for traffic blocking rules. Only one instance of this class is present in each CUDB node, which cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:

```
ManagedElement=1, CudbSystem=1, CudbLocalNode=<CUDB_Local_Node_Id>, CudbTrafficControlManager=1
```

Table 25 shows the attributes of the `CudbTrafficControlManager` class.

Table 25 Class `CudbTrafficControlManager`

Attribute Name	Data Type	Properties
<code>cudbTrafficControlManagerId</code> Identifies the instance of this class.	<code>EcimString</code> Range: 1 Value: 1	Mandatory Restricted



Table 25 Class *CudbTrafficControlManager*

Attribute Name	Data Type	Properties
adminState Defines the administrative state of the function.	BasicAdmState Available values: <ul style="list-style-type: none"> LOCKED: The resource is administratively prohibited from performing services for its users. UNLOCKED: The resource is administratively permitted to perform services for its users. This is independent of its inherent operability. Default value: LOCKED	Optional Read/Write
trafficControlManagerState Defines the operational state of the function. Shows if there is any inconsistency between the configuration data model and the node behavior.	OperState Available values: <ul style="list-style-type: none"> ENABLED: Node behavior is aligned with the configuration. DISABLED: There is a problem with activating the configuration change in the node. The node behavior may not be consistent with the configuration. 	Read only

2.4.13 Class *CudbTrafficBlockingRule*

The *CudbTrafficBlockingRule* class is used to block access to certain CUDB VIPs or services running on certain CUDB VIP ports. There are as many instances of this class as the number of VIP/ports to be blocked, which can be deleted from the configuration model.

The full path to the instances of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>,CudbTrafficControlManager=1,CudbTrafficBlockingRule=<CUDB_Traffic_Blocking_Rule_Id>
```

Table 26 shows the attributes of the *CudbTrafficBlockingRule* class.



Table 26 Class CudbTrafficBlockingRule

Attribute Name	Data Type	Properties
<code>cudbTrafficBlockingRuleId</code> Identifies the instance of this class.	<code>EcimString</code> Range: Integer, not zero. Example: 1	Mandatory Restricted
<code>blockedVIP</code> IP address which will be blocked.	<code>IpDNSAddress</code> Constraint: IPv4 or IPv6 address syntax, host names are not allowed. Example: 10.1.5.15 or 2001:cdba:0000:0000:0000:0000:3257:9652	Mandatory Restricted

2.5 Node Object Structures

This section describes the Ericsson Common Information Model (ECIM) structures that are part of the CUDB node configuration data model. ECIM structures contain attributes of different data types grouped together, and can be used for a variety of purposes.

2.5.1 Structure CudbAsyncActionProgress

The `CudbAsyncActionProgress` structure is used to show the status of an asynchronous administrative operation. Table 27 shows the members of the `CudbAsyncActionProgress` structure.

Table 27 Structure CudbAsyncActionProgress

Attribute Name	Data Type	Properties
<code>id</code> Identifies the instance of this class.	<code>EcimString</code> Range: 1 Example: 0	Mandatory Restricted
<code>actionId</code> Uniquely identifies the invocation of an action.	<code>EcimUint32</code> Example: 0	Read only
<code>actionName</code> Name of the invoked asynchronous action.	<code>EcimString</code> Example: <code>applyConfig</code>	Read only
<code>additionalInfo</code> Used for logging significant information.	<code>EcimString</code> Example: <code>applyConfig</code> automatically makes any configuration model change persistent.	Read only
<code>progressInfo</code> Textual information that describes the current state of the action execution.	<code>EcimString</code> Example: <code>applyConfig</code> execution running.	Read only



Table 27 *Structure CudbAsyncActionProgress*

Attribute Name	Data Type	Properties
progressPercentage Progress of the action.	EcimUInt32 Range: 0–100 Example: 10	Read only
result Result state of a completed action.	EcimEnumeration Possible values: SUCCESS, FAILURE, NOT_AVAILABLE Example: NOT_AVAILABLE	Read only
resultInfo Textual description of the outcome or result of the action.	EcimString Example: Ready.	Read only
state Current state of the action.	EcimEnumeration Possible values: CANCELLING, RUNNING, FINISHED, CANCELLED Example: RUNNING	Read only
timeActionStarted Date and time when the current action was started.	EcimString Example: 2016-07-05 10:52:16	Read only
timeActionCompleted Date and time when the action was completed (successfully or unsuccessfully).	EcimString Example: 2016-07-05 10:55:16	Read only
timeOfLastStatusUpdate Date and time when the <code>state</code> attribute of the structure was last updated.	EcimString Example: 2016-07-05 10:55:16	Read only

2.6 LDAP Access Object Classes

This section describes the classes for configuration of the LDAP access in the CUDB system.

2.6.1 Class CudbLdapAccess

The `CudbLdapAccess` class has one instance per CUDB node, which contains the CUDB-related LDAP access configuration of the local node. There is one instance per CUDB node, which cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:

```
ManagedElement=1, CudbSystem=1, CudbLocalNode=<CUDB_Local_Node_Id>, CudbLdapAccess=1
```

Table 28 shows the attributes of the `CudbLdapAccess` class.



Table 28 Class CudbLdapAccess

Attribute Name	Data Type	Properties
<code>cudbLdapAccessId</code> Identifies the instance of this class.	<code>EcimString</code> Range: 1 Value: 1	Mandatory Restricted
<code>cudbRootEntryDn</code> DN for the main directory entry in CUDB DIT. It is recommended to use the shortest root entry possible as it affects the total system dimensioning, specially when it comes to large CUDB deployments.	<code>EcimString</code> Syntax: DN format Constraint: It must be set to the same value in all CUDB nodes defined in a CUDB system. The values for the attributes in the DN must be in normalized form: <ul style="list-style-type: none">• If the type of a <code>rootDN</code> naming attribute is defined as case insensitive (has <code>caseIgnoreMatch</code> value in the <code>EQUALITY</code> property of the attribute definition in the LDAP schema), its value must be specified in lowercase.• If the type of a <code>rootDN</code> naming attribute is defined as case sensitive in the LDAP schema, the value does not need to be specified in lowercase, but be aware that the entered value will be considered as the normalized form to be checked when processing LDAP operations. Examples: <ul style="list-style-type: none">• "dc=telco-op,dc=com" Note: The <code>dc</code> is defined with <code>EQUALITY</code> set to <code>caseIgnoreMatch</code> in the schema. <ul style="list-style-type: none">• "attr=Telco_OP,dc=com" Note: The <code>attr</code> is defined as case sensitive in the schema and <code>Telco_OP</code> is the value passed in the LDAP operations to read or write entries.	Mandatory Restricted
<code>customDistributionPolicyEnabled</code> Specifies if a custom distribution policy library is loaded. When this attribute is changed to <code>true</code> , the library is loaded and when changed to <code>false</code> , the library is unloaded. For more information about distribution algorithms, refer to <i>CUDB LDAP Data Access</i> , Reference [4].	<code>EcimBoolean</code> Default value: <code>false</code>	Optional Read/Write



Table 28 Class CudbLdapAccess

Attribute Name	Data Type	Properties
<p>ldapAttrIndexes</p> <p>List of LDAP attributes (defined in some of the LDAP schemes managed in the CUDB system on the LDAP Access level) to be managed as searching indexes.</p> <p>For more information, refer to <i>CUDB Application Integration Guide</i>, Reference [13].</p>	<p>EcimString</p> <p>Syntax: LDAP attribute</p> <p>Constraint: When modifying this attribute, only adding new indexes is allowed. Removing existing indexes is not allowed. LdapAttrIndexes must be defined in the same order on every node.</p> <p>Example: [MSISDN, IMSICHO, IMSI]</p>	<p>Optional</p> <p>Read/Write</p> <p>multivalued</p>
<p>ldapRootPassword</p> <p>Password of the LDAP rootdn user.</p>	<p>EcimPassphraseString</p> <p>Default value: *****</p> <p>Constraint: This attribute can contain only ASCII alphabetic characters, numeric digit characters, and the following symbols: , - % = ? + ~ _</p> <p>For more information, refer to <i>CUDB Users and Passwords</i>, Reference [18].</p>	<p>Optional</p> <p>Read/Write</p>
<p>nodeLdapAuth</p> <p>Determines if the password of the LDAP users is either stored in clear text or hashed. If the value is SASL then the password is stored in clear text. If the value is SIMPLE the password is stored using a hash determined by the nodeLdapHash attribute</p> <p>The effect of this parameter can be overridden for specific LDAP users by setting a value in the userLdapAuth attribute in the corresponding CudbLdapUser instance. This parameter does not apply to the LDAP root user whose password is always stored encrypted.</p> <p>For more information, refer to <i>CUDB Security and Privacy Management</i>, Reference [12].</p>	<p>EcimString</p> <p>Default value: SIMPLE</p> <p>Note: If the value is SIMPLE then the LDAP user will not be able to use Simple Authentication and Security Layer (SASL) authentication.</p>	<p>Optional</p> <p>Read/Write</p>
<p>nodeLdapHash</p> <p>Specifies the type of hash chosen to store the password when the nodeLdapAuth attribute is SIMPLE, otherwise this attribute is not applicable. The effect of this parameter can be overridden for specific LDAP users by setting a value in the userLdapHash attribute in the corresponding CudbLdapUser instance.</p> <p>For more information, refer to <i>CUDB Security and Privacy Management</i>, Reference [12].</p>	<p>EcimString</p> <p>Default value: SHA-256</p>	<p>Optional</p> <p>Read/Write</p>
<p>redundancyLevel</p> <p>Number of LDAP FEs which can be down without the CUDB node losing its required level of performance. Only authorized Ericsson personnel can modify this attribute. For more information, refer to <i>CUDB High Availability</i>, Reference [2].</p>	<p>NumericMaxInclusive255</p> <p>Example: 4</p>	<p>Mandatory</p> <p>Read/Write</p>



2.6.2 Class CudbLdapUsersMgmt

The `CudbLdapUsersMgmt` class contains of CUDB LDAP users and CUDB LDAP users groups. There is just one instance per CUDB node and it cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>,CudbLdapAccess=1,CudbLdapUsersMgmt=1
```

Table 29 shows the attributes of the `CudbLdapUsersMgmt` class.

Table 29 Class `CudbLdapUsersMgmt`

Attribute Name	Data Type	Properties
<code>cudbLdapUsersMgmtId</code> Identifies the instance of this class.	<code>EcimString</code> Range: 1 Value: 1	Mandatory Restricted

2.6.3 Class CudbLdapUserGroup

The `CudbLdapUserGroup` class is used to specify groups of CUDB LDAP users. Instance of this class can be deleted from the configuration model.

The full path to the instances of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>,CudbLdapAccess=1,CudbLdapUsersMgmt=1,CudbLdapUserGroup=<CUDB_Ldap_User_Group_Id>
```

Table 30 shows the attributes of the `CudbLdapUserGroup` class.

Table 30 Class `CudbLdapUserGroup`

Attribute Name	Data Type	Properties
<code>cudbLdapUserGroupId</code> Identifies an LDAP user group used to compose the LDAP Organizational Unit (OU) used as parent for LDAP users.	<code>EcimString</code> Example: group1	Mandatory Restricted

2.6.4 Class CudbLdapUser

The `CudbLdapUser` class is used to specify CUDB LDAP users. Instance of this class can be deleted from the configuration model.

The full path to the instances of this class is as follows:



ManagedElement=1, CudbSystem=1, CudbLocalNode=<CUDB_Local_Node_Id>, CudbLdapAccess=1, CudbLdapUsersMgmt=1, CudbLdapUser=<CUDB_Ldap_User_Id>

Table 31 shows the attributes of the CudbLdapUser class.

Table 31 Class CudbLdapUser

Attribute Name	Data Type	Properties
countersGroup Group to which user belongs to regarding Per-Application Group LDAP node counters. For more information, refer to <i>CUDB LDAP Data Access</i> , Reference [4].	NumericMaxInclusive4 Range: 0–4 Default value:0	Optional Read/Write
cudbLdapUserId Identifies the instance of this class and it corresponds to the name of the LDAP user.	EcimString Constraint: This attribute must be unique in the whole CUDB system. Constraint: If SASL authentication is to be used for this user, then this attribute must not contain upper case letters. For more information on SASL authentication, refer to <i>CUDB Security and Privacy Management</i> , Reference [12]. Example: admin1 Constraint: The prefix internal in the cudbLdapUserId value is reserved for Ericsson internal use.	Mandatory Restricted
cudbLdapViewId Identifier of CudbLdapView attached to the actual user. An LDAP view cannot be assigned to a provisioning or reprovisioning user.	EcimString Constraint: It must correspond to the value of the ldapViewName attribute of a CudbLdapView object.	Optional Read/Write
cudbUserGroup The group to which the user belongs. " " means that the LDAP user does not belong to any group.	EcimString Constraint: If not " ", it must be an existing cudbLdapUserGroupId Constraint: Users that require SASL authentication cannot belong to any group. For more information on SASL authentication, refer to <i>CUDB Security and Privacy Management</i> , Reference [12]. Example: " "	Mandatory Restricted
cudbUserPassword The password of the LDAP user. For more information, refer to <i>CUDB Security and Privacy Management</i> , Reference [12].	EcimPasswordString Constraint: cudbUserPassword cannot be an empty string. Example: 0admin1Pwd	Mandatory Read/Write



Table 31 Class CudbLdapUser

Attribute Name	Data Type	Properties
isProvisioningUser This parameter has effects on how LDAP update operations are treated for this user in symmetrical split situations. If a user is assigned to an LDAP View, it cannot become a provisioning user. For more information, refer to <i>CUDB High Availability</i> , Reference [2].	EcimBoolean Default value: false	Optional Read/Write
isReProvisioningUser This parameter is used by a PG user to send re-provisioning operations after a mastership change. If a user is assigned to an LDAP View, it cannot become a reprovisioning user. For more information, refer to <i>CUDB High Availability</i> , Reference [2].	EcimBoolean Example: true Default value: false	Optional Read/Write
localReadsDsReplicationDelayThreshold Defines the maximum threshold (in seconds) for replication delay value, which is used to determine if the slave replica is too far behind the master replica. If the slave replica is too far behind, the data will not be read locally. This applies only to users whose DS read mode is set to Local Preferred (LP).	EcimUInt32 Range: 10–7200	Optional Read/Write
overloadRejectionWeight Used to set the LDAP traffic priority under PL or DS overload for this user. The lower value for this parameter means higher priority and a lower rejection rate. For more information, refer to <i>CUDB LDAP Data Access</i> , Reference [4].	EcimUInt32 Range: 1–5 Default value: 1	Optional Read/Write
readModeInDS Used to determine which DSG replica is used for read LDAP requests when an access to a DSG is required. Following are the possible values: <ul style="list-style-type: none">• Master Always (MA): Read requests for DSG data are always sent to the master DS replica.• Master Preferred (MP): Read requests for DSG data are sent to the master DS replica if available, otherwise, the request is sent to any other available replica.• Local Preferred (LP): Read requests for DSG data are sent to the closest DS replica (closest meaning: first the one in the node that received the request if available, otherwise, any replica in the site where the node is hosted. In case the required replica is not available on the local site, the request is forwarded to the master replica. If none of the previous is available, any available replica in any site). See the end of this table for information on the supported value combinations. This parameter can be overridden for a particular search request by using LDAP ReadMode control. For more information, refer to <i>CUDB LDAP Interwork Description</i> , Reference [10].	EcimString	Mandatory Read/Write



Table 31 Class CudbLdapUser

Attribute Name	Data Type	Properties
<p><code>readModeInPL</code></p> <p>Used to determine which PLDB replica is used for read LDAP requests when an access to PLDB is required. Following are the possible values:</p> <ul style="list-style-type: none"> • Master Always (MA): Read requests to the PLDB are always sent to the master replica. • Master Preferred (MP): Read requests to the PLDB are sent to the master replica if available, otherwise, the request is sent to any other available replica. • Local Preferred (LP): Read requests to the PLDB are sent to the local PLDB replica. <p>See the end of this table for information on the supported value combinations.</p> <p>This parameter can be overridden for a particular search request by using LDAP ReadMode control. For more information, refer to <i>CUDB LDAP Interwork Description</i>, Reference [10].</p>	EcimString	Mandatory Read/Write
<p><code>userLdapAuth</code></p> <p>Determines if the password of the LDAP user is either stored in clear text or hashed. If the value is empty, the value of <code>userLdapAuth</code> in LDAP database is equal to <code>nodeLdapAuth</code> at the time when the user is created. If the value is <code>SASL</code> then the password is stored in clear text. If the value is <code>SIMPLE</code> the password is stored using a hash determined by the <code>userLdapHash</code> attribute in this ObjectClass, if present, or the <code>nodeLdapHash</code> attribute in the <code>CudbLdapAccess</code> instance.</p> <p>If the value for this attribute is set, it prevails over <code>nodeLdapAuth</code> specified in <code>CudbLdapAccess</code> class.</p> <p>For more information, refer to <i>CUDB Security and Privacy Management</i>, Reference [12].</p>	EcimString	Optional Read/Write
<p><code>userLdapHash</code></p> <p>Specifies type of hash chosen to store the password when the <code>userLdapAuth</code> attribute is set to <code>SIMPLE</code>, otherwise this attribute is not applicable. If the value is empty, the value of <code>userLdapHash</code> in LDAP database is equal to <code>nodeLdapHash</code> at the time when the user is created. If the value for this attribute is set, it prevails over <code>nodeLdapHash</code> specified in <code>CudbLdapAccess</code> instance.</p> <p>For more information, refer to <i>CUDB Security and Privacy Management</i>, Reference [12].</p>	EcimString	Optional Read/Write

**Note:**

CUDB supports the following value combinations for the `readModeInPL` and `readModeInDS` LDAP user attributes:

- `readModeInPL=LP` and `readModeInDS=MP` for traffic applications and application FEs (such as Home Subscriber Server (HSS) and Home Location Register (HLR)).
- `readModeInPL=MA` and `readModeInDS=MA` for provisioning applications and application FEs (such as the PG).
- `readModeInPL=LP` and `readModeInDS=LP` for traffic applications with local read, based on replica availability.

Note: The `readModeInDS=LP` can be configured only if the Deployment Flexibility Value Package is available.

For more information, refer to *CUDB LDAP Data Access*, Reference [4].

Also, when the `cudbUserPassword` attribute is set, the configured password keeps its value without any encryption until the `applyConfig` action is executed. After executing the `applyConfig` administrative operation, this value is shown as `*****`.

The LDAP Data Views function supports accessing stored data through customizable views.

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

2.6.5 Class `CudbLdapViewsMgmt`

The LDAP Data Views function supports accessing stored data through customizable views.

The `CudbLdapViewsMgmt` class is used to contain views. Only one instance of this class is present in each CUDB node, which is created at installation and cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>,CudbLdapAccess=1,CudbLdapViewsMgmt=1
```

Table 32 shows the attributes of the `CudbLdapViewsMgmt` class.

Table 32 Class `CudbLdapViewsMgmt`

Attribute Name	Data Type	Properties
<code>cudbLdapViewsMgmtId</code> Identifies the instance of this class.	<code>EcimString</code> Value: 1	Mandatory Restricted



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

2.6.6 Class CudbLdapView

The LDAP Data Views function supports accessing stored data through customizable views.

The `CudbLdapView` class contains the specific LDAP views.

The full path to the instances of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>,CudbLdapAccess=1,CudbLdapViewsMgmt=1,CudbLdapView=<CUDB_Ldap_View_Id>
```

Table 33 shows the attributes of the `CudbLdapView` class.

Table 33 Class `CudbLdapView`

Attribute Name	Data Type	Properties
<code>cudbLdapViewId</code> Identifies the instance of this class.	NumericString Constraint: This attribute must be unique in the whole CUDB system. Example: 1	Mandatory Restricted
<code>ldapViewName</code> Identifies the name of the LDAP View being used.	EcimString Constraint: This attribute must be unique in the whole CUDB system. Example: TestView	Mandatory Restricted

For more information on the LDAP Data Views function, refer to *CUDB LDAP Data Views*, Reference [16].

Note:

- The LDAP Data Views function can only be used if the Application Facilitator Value Package is available.
- It is recommended that each newly added LDAP View instance has `cudbLdapViewId` attribute set with a 100 increment from the last LDAP View addition. For more information, refer to the *Creating LDAP Data View in the Data Model* section of *CUDB LDAP Data Views Management*, Reference [17].

2.7 Notifications Object Classes

Notifications object classes provide information about notifications to external network entities, for example, HSS/Subscription Locator Function (SLF) FE and HLR, when certain data in the CUDB objects changes. External network



entity details, objects in the CUDB to be monitored, and the content of the notifications can be configured through these classes.

For more information, refer to *CUDB Notifications*, Reference [19].

2.7.1 Class CudbNotifications

The `CudbNotifications` class is the root element for the notifications application FE, which cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbNotifications=1
```

Table 34 shows the attributes of the `CudbNotifications` class.

Table 34 Class *CudbNotifications*

Attribute Name	Data Type	Properties
<code>cudbNotificationsId</code> Identifies the instance of this class.	<code>EcimString</code> Range: 1 Value: 1	Mandatory Restricted
<code>enabled</code> Specifies if notifications are to be sent to endpoints. If set to <code>true</code> , notifications are sent. If set to <code>false</code> , notifications are not sent, and monitoring of data is stopped.	<code>EcimBoolean</code> Default value: <code>true</code>	Optional Read/Write
<code>maxReattempts</code> The maximum number of retries to send a notification to an FE. The retries are done when there is a connection error or the FE is not responding. A value of 0 indicates a notification is sent to an FE only once.	<code>NumericMaxInclusive255</code> Default value: 3	Optional Read/Write
<code>retryTime</code> The base time (in milliseconds) between attempts to send a notification to an FE.	<code>NumericRangeInclusive255to3600000</code> Default value: 1000	Optional Read/Write

2.7.2 Class CudbNotificationEvent

The `CudbNotificationEvent` class specifies a notification event to be sent to an application FE when a monitored CUDB object class attribute changes its value.

The full path to the instances of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbNotifications=1,CudbNotificationEvent=<CUDB_Notification_Event_Id>
```

Table 35 shows the attributes of the `CudbNotificationEvent` class.



Table 35 Class CudbNotificationEvent

Attribute Name	Data Type	Properties
cudbNotificationEventId Identifies the instance of this class.	EcimString Example: 1	Mandatory Restricted
eventId Identifier of the notification type and the value is for internal use.	EcimString Example: SAE-HLR Constraint: It must be unique across all CudbNotificationEvent instances.	Mandatory Read/Write
notificationString The information to be included in the notification event. The value of this attribute is included in the field notificationEvent of the SOAP message. For more information, refer to <i>CUDB SOAP Interwork Description</i> , Reference [20].	EcimString Example: mobilityEvent	Optional Read/Write

2.7.3 Class CudbNotificationEndPoint

The CudbNotificationEndPoint class specifies a notification endpoint that receives the specified notification event.

The full path to the instances of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbNotifications=1,CudbNotificationEvent=<CUDB_Notification_Event_Id>,CudbNotificationEndPoint=<CUDB_Notification_End_Point_Id>
```

Table 36 shows the attributes of the CudbNotificationEndPoint class.

Table 36 Class CudbNotificationEndPoint

Attribute Name	Data Type	Properties
cudbNotificationEndPointId Identifies the instance of this class.	EcimString Example: 1	Mandatory Restricted
name A label for free use.	EcimString Example: Server1	Mandatory Read/Write
URI The Uniform Resource Identifier (URI) of the endpoint that is to receive the notification event.	EcimString Syntax: URI format Example for IPv4 endpoint address: https://127.0.0.1:8080 Example for IPv6 endpoint address: https://[2001:1b70:8294:1995::199]:8080	Mandatory Read/Write



Table 36 Class CudbNotificationEndPoint

Attribute Name	Data Type	Properties
webService This attribute is concatenated to the attribute URI.	EcimString Example: /	Optional Read/Write
weight Used by the round-robin selection of an application FE to receive a notification event. The higher the value, the higher the weight of this endpoint in the round-robin selection. A value of 0 indicates that the notification is always sent to the endpoint.	EcimUint32 Default value: 3	Optional Read/Write

2.7.4 Class CudbNotificationObjectClass

The `CudbNotificationObjectClass` class specifies a CUDB subscriber object class whose attributes are monitored.

The full path to the instances of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbNotifications=1,CudbNotificationEvent=<CUDB_Notification_Event_Id>,CudbNotificationObjectClass=<CUDB_Notification_Object_Class_Id>
```

Table 37 shows the attributes of the `CudbNotificationObjectClass` class.

Table 37 Class CudbNotificationObjectClass

Attribute Name	Data Type	Properties
cudbNotificationObjectClassId Identifies the instance of this class.	EcimString Example: 1	Mandatory Restricted
dn The DN of the entry. If the value of the <code>type</code> attribute is <code>monitor</code> or <code>monitorAll</code> , the DN can be configured by using Portable Operating System Interface (POSIX) extended regular expressions. Also, it can contain partial DN (below DE part) or full DN (using regular expression for subscriber ID or any other data). If the value of the <code>type</code> attribute is <code>check</code> or <code>related</code> , POSIX regular expressions cannot be used and only partial DN can be defined. The values for the attributes in the <code>dn</code> must be in normalized form, that means, if the attribute type is case insensitive (this is specified in the LDAP schema), the value of the attribute must be written in lower case letters. If the attribute is case sensitive, the value of the attribute must be written as it is provisioned in CUDB.	EcimString Syntax: DN format Constraints: The string must evaluate to a POSIX extended regular expression. Example: "EpsDynInfId=EpsDynInf,EpsStaInfId=EpsStaInf,serv=eps" Example: "(EpsDynInfId=.*,EpsStaInfId=EpsStaInf,serv=eps).*"	Mandatory Read/Write

Table 37 Class *CudbNotificationObjectClass*

Attribute Name	Data Type	Properties
name The name of the object class whose attributes are involved in the notification.	EcimString Example: EpsDynInf	Mandatory Read/Write
type Defines the type of the attributes below this instance. Following are the possible values: <ul style="list-style-type: none"> • monitor: The data is to be monitored for change. Any change in the values of the attributes below this instance can trigger a notification except in case the attribute is a single value and the object class to which the attribute belongs is added or deleted in the same LDAP operation that is changing the value of the attribute. All instances of this class type under the same <i>CudbNotificationEvent</i> instance must have the same value for the <i>dn</i> attribute. • monitorAll: The data is to be monitored for change. Any change in the attributes below this instance can trigger a notification. This value can be used to monitor the creation or deletion of entries with DN matching the one configured in this object if the attribute in the leftmost RDN of the DN is configured below this object. • check: The data must be compared against a specified value to decide if a notification is to be sent. There can be more than one check item, in which case the notification is sent if any check item matches its comparison value. • related: The data is relevant to the notification event. The present value of the data is included in the notification. 	EcimString Example: monitor	Mandatory Read/Write

2.7.5 Class *CudbNotificationAttr*

The *CudbNotificationAttr* class specifies an attribute of the CUDB object class involved in the notification.

The full path to the instances of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbNotifications=1,CudbNotificationEvent=<CUDB_Notification_Event_Id>,CudbNotificationObjectClass=<CUDB_Notification_Object_Class_Id>,CudbNotificationAttr=<CUDB_Notification_Attr_Id>
```

Table 38 shows the attributes of the *CudbNotificationAttr* class.

Table 38 Class *CudbNotificationAttr*

Attribute Name	Data Type	Properties
cudbNotificationAttrId Identifies the instance of this class as defined in the corresponding LDAP schema.	EcimString Example: 1	Mandatory Restricted
name The name of the attribute of the CUDB object class. Only attributes of type <i>EcimString</i> and <i>EcimUint32</i> are supported.	EcimString Example: PSLOC	Mandatory Read/Write

**Table 38** Class *CudbNotificationAttr*

Attribute Name	Data Type	Properties
send When set to <code>true</code> , the attribute is sent in the notification. When set to <code>false</code> , the attribute is not sent in the notification.	<code>EcimBoolean</code> Example: <code>false</code>	Mandatory Read/Write
value The value to be used in the comparison with the current attribute value when the instance is below a <code>CudbNotificationObjectClass</code> instance of type <code>check</code> . The attributes must be encoded as follows: <ul style="list-style-type: none">• Character strings for non-binary attributes.• Base64-encoded strings with 2040 characters maximum length for binary attributes.	<code>EcimString</code> Example: <code>5</code>	Optional Read/Write

2.8 PG Object Classes

PG object classes provide information about PG endpoints, defined for configuration, related to Provisioning Assurance feature. PG endpoints access data and URLs can be configured through these classes.

For more information, refer to *CUDB High Availability*, Reference [2] and *CUDB LDAP Data Access*, Reference [4].

2.8.1 Class *CudbProvisioningGatewayMgmt*

The `CudbProvisioningGatewayMgmt` class is used to contain the classes that specify PG endpoint configuration. Only one instance of this class is present in each CUDB node, which cannot be deleted from the configuration model.

The full path to the instance of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbProvisioningGatewayMgmt=1
```

Table 39 shows the attributes of the `CudbProvisioningGatewayMgmt` class.

Table 39 Class *CudbProvisioningGatewayMgmt*

Attribute Name	Data Type	Properties
cudbProvisioningGatewayMgmtId Container class that contains the PGy endpoint instances.	<code>EcimString</code> Example: <code>1</code>	Mandatory Restricted



2.8.2 Class CudbProvGatewayEndPoint

The `CudbProvGatewayEndPoint` class specifies a PG endpoint that receives the reprovisioning request for the Provisioning Assurance after CUDB Mastership Change function. The number of `CudbProvGatewayEndPoint` instances equals the number of PGs existing in the UDC system. These instances can be deleted from the configuration model.

The full path to the instances of this class is as follows:

```
ManagedElement=1,CudbSystem=1,CudbProvisioningGatewayMgmt=1,CudbProvGatewayEndPoint=<CUDB_Prov_Gateway_End_Point_Id>
```

Table 40 shows the attributes of the `CudbProvGatewayEndPoint` class.

Table 40 Class CudbProvGatewayEndPoint

Attribute Name	Data Type	Properties
<code>cudbProvGatewayEndPointId</code> PG endpoint definition	NumericString Range: 1-255 Example: 1	Mandatory Read/Write
<code>user</code> User to connect with PG through http.	EcimString Example: pgHttp1	Mandatory Read/Write
<code>password</code> Password to connect with PG through http.	EcimPasswordString Example: 0pgHttp1Pwd	Mandatory Read/Write



Table 40 Class CudbProvGatewayEndPoint

Attribute Name	Data Type	Properties
<code>replayRequestURL</code> Universal Resource Locator (URL) in URI format to send to the PG a request to start re-provisioning.	<code>EcimString</code> Syntax: URI format as follows: <code>http://<PG OAM_VIP>:<replay port>/<replay URL suffix></code> or <code>http:// [<PG OAM_VIP>] :<replay port>/<replay URL suffix></code> in case of IPv6 PG OAM_VIP address. Example for IPv4: <code>http:// 10.250.2.139:8282/re player/execute</code> Example for IPv6: <code>http:// [2001:a234::2002]:82 82/replayer/execute</code>	Mandatory Read/Write
<code>replayStatusURL</code> URL in URI format to ask to the PG the status of the re-provisioning.	<code>EcimString</code> Syntax: URI format as follows: <code>http://<PG OAM_VIP>:<status port>/<status URL suffix></code> or <code>http:// [<PG OAM_VIP>] :<status port>/<status URL suffix></code> in case of IPv6 PG OAM_VIP address. Example for IPv4: <code>http:// 10.250.2.139:8282/re player/state</code> Example for IPv6: <code>http:// [2001:a234::2002]:82 82/replayer/state</code>	Mandatory Read/Write

2.9 CUDB Administrative Operations

This section describes the administrative operations available in the CUDB system. See Table 41 for the list of available administrative operations, and the below subsections for more information on them.

Table 41 Administrative Operations

Class	Administrative Operation	Command Options	Execution Type
<code>CudbLocalNode</code>	<code>applyConfig</code> Administrative operation used for activating configuration changes.	N/A	Asynchronous
<code>CudbLocalNode</code>	<code>updateUserInfo</code> Administrative operation used to update the local node configuration with the last changes of LDAP users in the CUDB node where the command is executed.	N/A	Asynchronous



2.9.1 **applyConfig**

The `applyConfig` administrative operation analyzes the committed configuration changes, and automatically triggers various actions to apply them and make them persistent.

If any of these actions fail, then the whole command fails. A log of the performed actions is located in the system log of the System Controller (SC) where `applyConfig` is invoked.

The attribute `applyConfigStatus`, located in the class `CudbLocalNode`, contains information about the current state of the `applyConfig` execution. See Section 2.4.1 on page 20 for more information about `applyConfigStatus`.

If the execution of the `applyConfig` administrative operation is successful, it can be assumed that the requested configuration changes, performed either through the CUDB configuration Command Line Interface (CLI) session or the NETCONF interface, are effective, and are persistent, that is, the changes remain effective even after a restart.

If the execution of `applyConfig` fails, the state of the CUDB node becomes inconsistent with the requested configuration changes, even if those changes still appear in the configuration model.

Executed actions cannot be automatically undone. The result of some of these actions can permanently affect the state of CUDB node, while others have impersistent effect on the state of the CUDB node (that is, such changes are undone upon a restart). Certain required actions may not even be executed at all in case a problem is encountered.

The execution of `applyConfig` is asynchronous. Once `applyConfig` is invoked successfully, the CLI or NETCONF console is returned to the user.

Warning!

Do not perform configuration changes while `applyConfig` is running.

2.9.1.1 **Requisites**

Before invoking `applyConfig`, check the `applyConfigStatus` attribute to make sure that the current state of the action is not `RUNNING`.

2.9.1.2 **Input Parameters**

Not available.



2.9.1.3 Output

Once `applyConfig` is invoked, one of the following messages is logged in the console:

- `applyConfig invoked successfully.`
- `applyConfig invocation failed. cudbApplyConfig.lock exists. Another instance is running.`
- `applyConfig invocation failed because cudbSwBackup is running. cudbSwBackup.lock file exists.`
- `applyConfig invocation failed. Configuration file generation did not finish successfully.`
- `Failed to update applyConfigStatus.`

2.9.1.4 Common Issues

Invocation of the `applyConfig` administrative operation can fail for one of the following reasons:

- `applyConfig` cannot be started if another instance is running. In this case, COM will return the following message:

```
applyConfig invocation failed. cudbApplyConfig.lock
exists. Another instance is running.
```

Wait until the process has finished before performing any new configuration model changes. The status of the current execution can be checked under `applyConfigStatus` in the `CudbLocalNode` class.

- `applyConfig` cannot be started if the software and configuration backup procedure is running. In this case, COM will return the following message:

```
applyConfig invocation failed because cudbSwBackup is
running. cudbSwBackup.lock file exists.
```

Wait until the backup process has finished before performing any new configuration model changes.

After the execution of the `applyConfig` administrative operation, check the result in the `applyConfigStatus` attribute. If its `state` attribute is set to `FINISHED` and the `result` attribute does not show `SUCCESS`, it is recommended to solve the problems reported in the `resultInfo` attribute of the `applyConfigStatus` as soon as possible. To analyze reported problems, check the system log of the SC where the active instance of the CUDB Object Implementer component is running. Then execute `applyConfig` again.

To find the active instance of the CUDB Object Implementer component, use the following command:



```
cudbHaState | grep ERIC-CUDB_CUDBOI
```

Repeat this cycle until either of the following circumstances occur:

- Administrative operation `applyConfig` is executed successfully.
- Administrative operation `applyConfig` fails, and there is no possible or known method of recovering from the reported problem, then perform a software restore from backup.

Repeat the configuration transaction from the beginning after the restore, and if the problem persists, contact the next level of support.

Note: In case of an error not described in this section, contact the next level of support.

See the Object Model Modification Procedure in Section 4.2 on page 63 for more information on how to check value of the `applyConfigStatus` parameter.

2.9.1.5 Examples of Use

See the below procedures for examples of using `applyConfig`:

Example 1

1. Establish a CUDB CLI session towards the CUDB node.
2. Access the data model by establishing a CUDB configuration CLI session in the active SC with the following command:

```
/opt/com/bin/cliss
```

See Step 3 in Section 4.2.1 on page 63 for more information on how to find the active SC.

3. Execute the following command to invoke the `applyConfig` administrative operation:

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<node_id>,ap  
plyConfig
```

Example 2

1. Establish a NETCONF session towards the CUDB node. For more information on how to establish a NETCONF session, refer to *CUDB System Administrator Guide*, Reference [21].
2. Execute the following command to invoke the `updateUserInfo` administrative operation:

```
<?xml version="1.0" encoding="UTF-8"?>  
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  
  <capabilities>
```



```
<capability>urn:ietf:params:netconf:base:1.0</capability>
</capabilities>
</hello>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:com:ericsson:ecim:1.0">
    <data>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
        <managedElementId>1</managedElementId>
        <CudbSystem xmlns="urn:com:ericsson:ecim:CUDB">
          <cudbSystemId>1</cudbSystemId>
          <CudbLocalNode xmlns="urn:com:ericsson:ecim:CUDB">
            <cudbLocalNodeId>[Cudb_Local_node_Id]</cudbLocalNodeId>
            <applyConfig>
              </applyConfig>
            </CudbLocalNode>
          </CudbSystem>
        </ManagedElement>
      </data>
    </action>
  </rpc>
]]>]]>
```

where [Cudb_Local_node_Id] is the ID of the local CUDB node.

If `applyConfig` is invoked successfully, the `applyConfigStatus` attribute is updated when the execution starts and finishes. Table 42 shows the possible updates of `applyConfigStatus`, based on the states of the `applyConfig` execution.

Table 42 *applyConfig States and Results*

applyConfig State	Running	Finished Successfully	Finished with Warnings	Finished with Errors
Update of <code>applyConfig</code> attribute.	<code>state=RUNNING</code> <code>result=NOT_AVAILABLE</code>	<code>state=FINISHED</code> <code>result=SUCCESS</code> <code>resultInfo=Ready.</code>	<code>state=FINISHED</code> <code>result=SUCCESS</code> <code>resultInfo=<information_about_warning>: Ready.</code>	<code>state=FINISHED</code> <code>result=SUCCESS</code> <code>resultInfo=<information_about_error></code>

Note: If the value of the `result` attribute is `FAILURE`, check the `resultInfo` attribute for more information about the error.

2.9.2 updateUserInfo

The `updateUserInfo` administrative operation updates the local node configuration with the last changes of LDAP users from the CUDB node where the changes are performed. A log of the performed actions is located in the system log of the SC where the `updateUserInfo` is invoked.

The `updateUserInfoStatus` attribute, located in the `CudbLocalNode` class, contains information about the current state of the `updateUserInfo` execution. For more information about the `updateUserInfoStatus` attribute, see Section 2.4.1 on page 20.

If the execution of an `updateUserInfo` administrative operation is successful, it can be assumed that the requested configuration changes are effective and consistent.



If `updateUserInfo` fails, the information about LDAP users on the CUDB nodes on the system remain inconsistent until the problem is solved and `updateUserInfo` is executed successfully.

Execution of `updateUserInfo` is asynchronous. Once `updateUserInfo` is invoked successfully, the CLI or NETCONF console is returned to the user.

Warning!

Do not perform configuration changes while `applyConfig` is running.

2.9.2.1 Requisites

Before invoking `updateUserInfo`, check the `updateUserInfoStatus` attribute to make sure that the current state of the action is not `RUNNING`.

2.9.2.2 Input Parameters

Not available.

2.9.2.3 Output

Once the `updateUserInfo` is invoked, one of the following messages is displayed on the console:

- `updateUserInfo` invoked successfully.
- `updateUserInfo` invocation failed. Another instance is running.
- Failed to update `updateUserInfoStatus`.

2.9.2.4 Common Issues

The `updateUserInfo` operation cannot be started if another instance is already running. In this case, COM will return the following message:

```
updateUserInfo invocation failed. Another instance is running.
```

Wait until the process is finished before performing any new configuration model changes. The status of the current execution can be checked under the `updateUserInfoStatus` attribute in the `CudbLocalNode` class.

After the execution of the `updateUserInfo` administrative operation, check the result in `updateUserInfoStatus`. If its state attribute is set to `FINISHED` and the result attribute does not show `SUCCESS`, it is



recommended to solve the problems reported in the `resultInfo` attribute of the `updateUserInfoStatus` as soon as possible. To analyze reported problems, check the system log of the SC where the active instance of the CUDB Object Implementer component is running, then execute `applyConfig` again.

To find the active instance of the CUDB Object Implementer component, use the following command:

```
cudbHaState | grep ERIC-CUDB_CUDBOI
```

In case the `updateUserInfoStatus` state is set to `FINISHED`, the `result` attribute shows `FAILURE` and the `resultInfo` reports the successful update of LDAP users and groups, but an error with configuration file generation appears, contact the next level of support.

2.9.2.5 Examples of Use

See the below procedures for examples of using `updateUserInfo`:

Example 1: Executing `updateUserInfo` through CUDB CLI

1. Establish a CUDB CLI session towards the CUDB node.
2. Access the data model by establishing a CUDB Configuration CLI session in the active SC with the following command:

```
/opt/com/bin/cliss
```

See Step 3 in Section 4.2.1 on page 63 for more information on how to find the active SC.

3. Execute the following command to invoke the `updateUserInfo` administrative operation:

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<node_id>,updateUserInfo
```

Example 2: Executing `updateUserInfo` with NETCONF

1. Establish a NETCONF session towards the CUDB node. For more information on how to establish a NETCONF session, refer to *CUDB System Administrator Guide*, Reference [21].
2. Execute the following command to invoke the `updateUserInfo` administrative operation:

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
```



```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:com:ericsson:ecim:1.0">
    <data>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
        <managedElementId>1</managedElementId>
        <CudbSystem xmlns="urn:com:ericsson:ecim:CUDB">
          <cudbSystemId>1</cudbSystemId>
          <CudbLocalNode xmlns="urn:com:ericsson:ecim:CUDB">
            <cudbLocalNodeId>[Cudb_Local_node_Id]</cudbLocalNodeId>
            <updateUserInfo>
              </updateUserInfo>
            </CudbLocalNode>
          </CudbSystem>
        </ManagedElement>
      </data>
    </action>
  </rpc>
</></>
```

where [Cudb_Local_node_Id] is the ID of the local CUDB node.

If the updateUserInfo is invoked successfully, the updateUserInfoStatus attribute is updated when execution starts and finishes. Table 43 shows the update of the updateUserInfoStatus attribute based on the state of updateUserInfo execution.

Table 43 updateUserInfo State and Corresponding Updates of updateUserInfoStatus Attribute

updateUserInfo State	Running	Finished Successfully	Finished with Errors
Update of updateUserInfo Status attribute.	state=RUNNING result=NOT_AVAILABLE	state=FINISHED result=SUCCESS resultInfo=Ready.	state=FINISHED result=FAILURE resultInfo=<information_about_error>

Note: If the value of the result attribute is FAILURE, check the resultInfo attribute for more information about the error.

2.9.3 cancelApplyConfig

The cancelApplyConfig administrative operation, which is restricted to Ericsson personnel only, is used for canceling a hanging applyConfig execution.





3 Initial Configuration

All the instances have **ManagedElement=1** as base.

3.1 Considerations

CUDB configuration is made on a per CUDB node basis. So, when configuring the complete system, the following considerations must be taken:

- There is no Domain Name System (DNS) resolution in CUDB which implies that each external network element, that needs to be configured, must be referred to with an IP address and not a domain name.
- CUDB node identifications must be coherent in the whole CUDB system. It means that a CUDB node identifier is not repeated either in two `CudbLocalNode` classes in different CUDB nodes, or in two `CudbLocalNode` classes, or in `CudbRemoteNode` in the same CUDB node. A CUDB node has the same identifier as a `CudbLocalNode` in one CUDB node, and as a `CudbRemoteNode` in the rest of CUDB nodes in the rest of the CUDB system.
- PLDB replicas must have a different `instancePriority` attribute across the whole CUDB system.
- DS replicas belonging to same DSG must have a different `instancePriority` attribute across the whole CUDB system.
- Attribute `cudbRootEntryDn`, specifying DN for the main directory entry in CUDB DIT, must be equal in **all** CUDB nodes across the CUDB system.
- Attribute `ldapAppSrvSchema` must be equal in **all** CUDB nodes across the CUDB system.
- All SQL schema related attributes, `sqlAppSrvPlSchema` and `sqlAppSrvDsSchema`, must be equal in **all** CUDB nodes across the CUDB system.
- When including new instances belonging to `CudbLocalPl` or `CudbLocalDs` classes in a CUDB node, `CudbRemotePl` and `CudbRemoteDs` instances, belonging to the `cudbRemoteNode` instance corresponding to the CUDB node where the physical clusters have been created, must be added in **all** the other CUDB nodes in the CUDB system with the same information.





4 Configuration Modification Procedure

This section describes the procedure to modify the existing configuration of the CUDB system. This procedure includes adding, deleting, and modifying attributes.

4.1 Preconditions

A user with write privileges must exist to view existing data values and also to change the data. For further information about credentials, refer to *CUDB Users and Passwords*, Reference [18].

4.2 Object Model Modification Procedure

Warning!

It is recommended to perform all configuration changes in the maintenance window period.

4.2.1 Modification Procedure Using CUDB Configuration CLI

All parameters in the MOM are accessible with the CUDB configuration CLI. For more information about CLI, refer to *CUDB System Administrator Guide*, Reference [21]. The steps of the object model modification procedure are the following:

1. Establish a CUDB CLI session towards the CUDB node by executing the following command:


```
ssh -l cudbadmin <CUDB_Node_OAM_IP_Address>.
```
2. If there is no backup of the present configuration, perform the backup by executing the `cmw-configuration-persist` command. If the backup is already created, proceed to Step 3.
3. Establish a CUDB configuration CLI session in the active SC. Refer to the “Finding the Active System Controller Blade” section of *CUDB System Administrator Guide*, Reference [21] for more information on how to find the active SC.



4. Make sure that all previous configuration changes have been successfully applied in the node. To do **show**, check the changes with the following command:

```
show ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>,applyConfigStatus
```

The value of the `state` attribute must be `FINISHED`, while the value of the `result` attribute must be `SUCCESS`.

5. Set the configuration session by executing the **configure** command.
6. Execute the **ManagedElement=1,CudbSystem=1, ... <value>** command to change the CUDB configuration model. Then, use the **commit** command to commit the changes. See the below examples for more information on how to perform configuration changes:

- Example 1: To change the value of the attribute `enabled` in the class `CudbLocalDs`, execute the following commands:

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>,CudbLocalDs=<CUDB_Local_Ds_Id>
```

```
enabled=<value>
```

```
commit
```

In the above command, `<CUDB_Local_Node_Id>` is ID of the local CUDB node. `<CUDB_Local_Ds_Id>` is the ID of the specific instance of the local DSG. Finally, `<value>` is the value of the `enabled` attribute which can be either `true` or `false`.

- Example 2: Instead of writing the full path (as done in Example 1), it is possible to navigate to the element being modified while being in `config` mode. In that case, the same example can be performed like shown below:

```
>configure
```

```
(config)>ManagedElement=1
```

```
(config-ManagedElement=1)>CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>
```

```
(config-CudbLocalNode=<CUDB_Local_Node_Id>)>CudbLocalDs=<CUDB_Local_Ds_Id>
```

```
(config-CudbLocalDs=<CUDB_Local_Ds_Id>)>enabled=true
```

```
(config-CudbLocalDs=<CUDB_Local_Ds_Id>)>commit
```

- Example 3: To add a new instance of the `CudbDsGroup` class to the configuration model, execute the following commands:



```

ManagedElement=1,CudbSystem=1,CudbDsGroup=<CUDB_Ds_Group_Id>

memoryEligibleThreshold=<Memory_Eligible_Threshold_Value>

memoryWarningThreshold=<Memory_Warning_Threshold_Value>

masterReplicationChannel1Port=<Master_Replication_Channel1_Port_Value>

masterReplicationChannel2Port=<Master_Replication_Channel2_Port_Value>

accessPort=<Access_Port_Value>

commit

```

Note: The value of <CUDB_Ds_Group_Id> must be unique, and must be assigned to the `cudbDsGroupId` attribute of the `CudbDsGroup` class. The same applies when adding a new instance of any class in the model.

All operations are executed as a unique transaction.

7. If any objects or attribute values must be deleted, then execute the `no ManagedElement=1,CudbSystem=1, ... <objectClass>=<objectId>` command to delete objects or attribute values. Then, use the `commit` command to commit the changes. See the below examples for more information on how to delete objects and attribute values in the configuration model:
 - Example 4: To delete a specific instance of the `CudbNotificationEndPoint` class, modify the configuration model as shown below:

```

no ManagedElement=1,CudbSystem=1,CudbNotifications=1,
CudbNotificationEvent=1,CudbNotificationEndPoint=<CUDB_Notification_End_Point_Id>

commit

```

- Example 5: To delete the value of the `userLabel` attribute of the `CudbSystem` class, execute the following commands:

```

no ManagedElement=1,CudbSystem=1,userLabel

commit

```

All operations are executed as a unique transaction.

Note: To delete a parent class, first delete all of its subclasses.



8. Commit the changes by executing the `commit` command. By default, after each `commit`, the CUDB CLI exits from configuration mode. Execute `commit -s` to stay in configure mode after committing changes.
9. Check log files to see the result of the operations. For more information, refer to *CUDB Node Logging Events*, Reference [22].
10. Check configuration changes by executing the `show ManagedElement=1,CudbSystem=1, ...` command. See the below examples for more information on how to check configuration changes.

Note: Remember to use `show verbose` instead of `show` for not mandatory attributes that must not have set value, or for optional attributes whose value is set to the default one. To see all attributes of specific class as well as its whole subtree, use `show all`.

- Example 6: To check all attributes of specific instance of the `CudbDsGroup` class, execute the following command:

```
show verbose ManagedElement=1,CudbSystem=1,CudbDsGroup=<CUDB_Ds_Group_Id>
```

To check the attributes of the `CudbLocalNode` class and its whole subtree, execute the following command:

```
show all ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>
```

11. Activate the configuration changes in the CUDB node by executing the `applyConfig` administrative operation with the following command:

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_Node_Id>,applyConfig
```

If the `state` attribute is set to `FINISHED` but the value of the `result` attribute is not `SUCCESS`, contact the next level of support.

12. Exit the CUDB configuration CLI console by executing the `exit` command.
13. Exit the CUDB CLI session by executing the `exit` command.

Warning!

Always use only one `commit` command to commit changes, and then one `applyConfig` administrative operation to activate the configuration changes. Avoid using several commits followed by one single `applyConfig` execution.



4.2.2 Modification Procedure Using NETCONF

Perform the following steps to modify the CUDB configuration model through NETCONF:

1. Open the NETCONF client and establish a NETCONF session towards the CUDB node. Refer to *CUDB System Administrator Guide*, Reference [21] for more information about NETCONF.
2. Change or fetch the configuration using the NETCONF client. The CUDB NETCONF interface is configured to use commit at `<close-session>` commit behavior. This means that changes are committed only after the session is closed. The configuration is changed by sending NETCONF commands in XML form, either directly or using a GUI client such as the Ericsson Netconf Browser. To check the result of the commit command, send a `close-session` message. See Section 4.2.2.1 on page 68 for examples showing how to change the configuration model through NETCONF.
3. Activate the configuration changes in the CUDB node on an SC, as described in Step 11. To execute the `applyConfig` operation, use the following commands:

```
<?xml version="1.0" encoding="UTF-8"?>
  <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
      <capability>urn:ietf:params:netconf:base:1.0</capability>
    </capabilities>
  </hello>
</>>
<?xml version="1.0" encoding="UTF-8"?>
  <rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <action xmlns="urn:com:ericsson:ecim:1.0">

      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
        <managedElementId>1</managedElementId>
        <CudbSystem xmlns="urn:com:ericsson:ecim:CUDB">
          <cudbSystemId>1</cudbSystemId>
          <CudbLocalNode xmlns="urn:com:ericsson:ecim:CUDB">
            <cudbLocalNodeId>[CUDB_Local_Node_Id]</cudbLocalNodeId>
            <applyConfig>
              </applyConfig>
            </CudbLocalNode>
          </CudbSystem>
        </ManagedElement>

      </action>
    </rpc>
  </>>
```

To check the status of the `applyConfig` action, use the following command:

```
<?xml version="1.0" encoding="UTF-8"?>
  <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
      <capability>urn:ietf:params:netconf:base:1.0</capability>
    </capabilities>
  </hello>
</>>
<?xml version="1.0" encoding="UTF-8"?>
  <rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get xmlns="urn:com:ericsson:ecim:1.0">
      <filter>
        <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
```



```

        <managedElementId>1</managedElementId>
        <CudbSystem xmlns="urn:com:ericsson:ecim:CUDB">
          <cudbSystemId>1</cudbSystemId>
          <CudbLocalNode xmlns="urn:com:ericsson:ecim:CUDB">
            <cudbLocalNodeId>[CUDB_Local_Node_Id]</cudbLocalNodeId>
            <applyConfigStatus>
              </applyConfigStatus>
            </CudbLocalNode>
          </CudbSystem>
        </ManagedElement>
      </filter>
    </get>
  </rpc>
}>]]>

```

4.2.2.1 Changing the Configuration Model Through NETCONF

This section provides examples of changing the configuration model through NETCONF.

4.2.2.1.1 Change enabled Attribute in CudbLocalDs

To change the value of the attribute enabled in the class CudbLocalDs, modify the configuration model as follows.

1. Execute the following:

```

<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>
]>]]>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <default-operation>merge</default-operation>
    <config>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop" xmlns:ns2="urn:com:ericsson:ecim:CudbMOM">
        <managedElementId>1</managedElementId>
        <CudbSystem>
          <cudbSystemId>1</cudbSystemId>
          <CudbLocalNode>
            <cudbLocalNodeId>[CUDB_Local_Node_Id]</cudbLocalNodeId>
            <CudbLocalDs>
              <cudbLocalDsId>[CUDB_Local_Ds_Id]</cudbLocalDsId>
              <enabled>[value]</enabled>
            </CudbLocalDs>
          </CudbLocalNode>
        </CudbSystem>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>
]>]]>

```

In the XML above, [CUDB_Local_Node_Id] is the ID of the local CUDB node. [CUDB_Local_Ds_Id] is the ID of the specific instance of the local DSG. Finally, [value] is the value of the enabled attribute which can be either true or false.



2. To commit the above changes, send the close-session command with the below XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
  <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
      <capability>urn:ietf:params:netconf:base:1.0</capability>
    </capabilities>
  </hello>
]>]]>
  <rpc message-id="2" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <close-session/>
  </rpc>
]>]]>
```

3. If the commit is successful, the reply message looks like the below example:

```
<?xml version="1.0" encoding="UTF-8"?>
  <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
    <ok/>
  </rpc-reply>
```

If the commit fails, the reply message looks like the below example:

```
<?xml version="1.0" encoding="UTF-8"?>
  <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="5">
    <rpc-error>
      <error-type>application</error-type>
      <error-tag>operation-failed</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Transaction commit failed, [detailed error description]</error-message>
    </rpc-error>
  </rpc-reply>
]>]]>
```

4.2.2.1.2 Add a New Instance of CudbDsGroup

To add a new instance of CudbDsGroup, modify the configuration model as shown below.



```

<?xml version="1.0" encoding="UTF-8"?>
  <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
      <capability>urn:ietf:params:netconf:base:1.0</capability>
    </capabilities>
  </hello>
]>]]>
  <rpc message-id="3" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
      <target>
        <running/>
      </target>
      <default-operation>merge</default-operation>
      <config>
        <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop" xmlns:ns2="urn:com:ericsson:ecim:CudbMOM">
          <managedElementId>1</managedElementId>
          <CudbSystem>
            <cudbSystemId>1</cudbSystemId>
            <CudbDsGroup>
              <cudbDsGroupId>[CUDb_Ds_Group_Id]</cudbDsGroupId>
              <memoryWarningThreshold>[Memory_Warning_Threshold_Value]</memoryWarningThreshold>
              <memoryEligibleThreshold>[Memory_Eligibility_Threshold]</memoryEligibleThreshold>
              <masterReplicationChannel1Port>[Master_Replication_Channel1_Port_Value]</masterReplicationChannel1Port>
              <masterReplicationChannel2Port>[Master_Replication_Channel2_Port_Value]</masterReplicationChannel2Port>
              <accessPort>[Access_Port_Value]</accessPort>
            </CudbDsGroup>
          </CudbSystem>
        </ManagedElement>
      </config>
    </edit-config>
  </rpc>
]>]]>

<?xml version="1.0" encoding="UTF-8"?>
  <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
      <capability>urn:ietf:params:netconf:base:1.0</capability>
    </capabilities>
  </hello>
]>]]>
  <rpc message-id="4" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <close-session/>
  </rpc>
]>]]>

```

4.2.2.1.3 Delete a specific instance of CudbDsGroup

To delete a specific instance of the `CudbDsGroup` class, modify the configuration model as follows.

Note: To delete a parent class, first delete all of its subclasses.



```
<?xml version="1.0" encoding="UTF-8"?>
  <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
      <capability>urn:ietf:params:netconf:base:1.0</capability>
    </capabilities>
  </hello>
</>>
<rpc message-id="5" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <default-operation>merge</default-operation>
    <config>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop" xmlns:ns2="urn:com:ericsson:ecim:CudbMOM">
        <managedElementId>1</managedElementId>
        <CudbSystem>
          <cudbSystemId>1</cudbSystemId>
          <CudbDsGroup operation="delete">
            <cudbDsGroupId>[CUDb_Ds_Group_Id]</cudbDsGroupId>
          </CudbDsGroup>
        </CudbSystem>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>
</>>>

<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>
</>>
<rpc message-id="6" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <close-session/>
</rpc>
</>>>
```

4.2.2.1.4 Delete the Value of userLabel Attribute of CudbSystem Class

To delete the value of the `userLabel` attribute of the `CudbSystem` class, modify the configuration model as follows.



```
<?xml version="1.0" encoding="UTF-8"?>
  <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
      <capability>urn:ietf:params:netconf:base:1.0</capability>
    </capabilities>
  </hello>
]>]]>
  <rpc message-id="7" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
      <target>
        <running/>
      </target>
      <default-operation>merge</default-operation>
      <config>
        <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop" xmlns:ns2="urn:com:ericsson:ecim:CudbMOM">
          <managedElementId>1</managedElementId>
          <CudbSystem>
            <cudbSystemId>1</cudbSystemId>
            <userLabel operation="delete"/>
          </CudbSystem>
        </ManagedElement>
      </config>
    </edit-config>
  </rpc>
]>]]>

<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>
]]>]]>
<rpc message-id="8" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <close-session/>
</rpc>
]]>]]>
```



Glossary

For the terms, definitions, acronyms and abbreviations used in this document, refer to *CUDB Glossary of Terms and Acronyms*, Reference [23].





Reference List

CUDB Documents

- [1] *CUDB Multiple Geographical Areas*
- [2] *CUDB High Availability*
- [3] *CUDB Subscription Reallocation*
- [4] *CUDB LDAP Data Access*
- [5] *Storage Engine, Replication Delay Too High In DS*
- [6] *Storage Engine, Replication Delay Too High In PLDB*
- [7] *Storage Engine, High Load In DS*
- [8] *Storage Engine, High Load in PLDB*
- [9] *Storage Engine, Memory Usage Too High In PLDB, Warning*
- [10] *CUDB LDAP Interwork Description*
- [11] *CUDB Application Schema Update*
- [12] *CUDB Security and Privacy Management*
- [13] *CUDB Application Integration Guide*
- [14] *LDAP Front End, High Load in LDAP Processing Layer*
- [15] *CUDB Node Network Description*
- [16] *CUDB LDAP Data Views*
- [17] *CUDB LDAP Data Views Management*
- [18] *CUDB Users and Passwords, 3/00651-HDA 104 03/10*
- [19] *CUDB Notifications*
- [20] *CUDB SOAP Interwork Description*
- [21] *CUDB System Administrator Guide*
- [22] *CUDB Node Logging Events*
- [23] *CUDB Glossary of Terms and Acronyms*



Other Ericsson Documents

[24] *COM Management Guide*

Other Documents and Online References

[25] *IETF RFC 791* <https://tools.ietf.org/html/rfc791>

[26] *IETF RFC 4291* <https://tools.ietf.org/html/rfc4291>