ERICSSON

CUDB Node Configuration Data Model Description

CONFIGURATION MODEL

1/19202-CSH 109 067/10 Uen A

Copyright

© Ericsson AB 2016. 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 other trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.

Contents

1 Introduction

This document describes configuration model of Ericsson Centralized User Database (CUDB) and the configuration modification procedure. This document applies to all CUDB nodes into a CUDB system because all CUDB nodes are configured in the same way.

1.1 Scope

This document covers the following:

- 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 5: Updated description of Read only attributes in Table 1.
- Section 2.1 on page 5: Updated Table 2 with information on `EcimStruct`, `EcimAction`, and `EcimEnumeration` data types.
- Section 2.2 on page 6: Added new classes `CudbTrafficControlManager` and `CudbTrafficBlockingRule` in Figure 1.
- Section 2.3.3 on page 11: Added a Warning.
- Section 2.4.1 on page 18: Updated Table 14 with information on `applyConfigStatus` attribute.
- Section 2.4.1 on page 18, Section 2.4.2 on page 21, Section 2.4.3 on page 22, Section 2.4.8 on page 26, and Section 2.4.9 on page 28: Added new data type values to the corresponding attributes in tables.

- Section 2.4.2 on page 21: Updated `CudbLocalPl` class description.
- Section 2.4.3 on page 22: Updated `CudbLocalDs` class description and data type in Table 16.
- Section 2.4.6 on page 25 and Section 2.4.7 on page 25: Updated sections with reference to *CUDB Security and Privacy Management*, Reference [12].
- Section 2.4.9 on page 28: Updated `CudbRemotePl` class description.
- Section 2.4.10 on page 29: Updated data type in Table 23.
- Section 2.4.12 on page 31: Added section on the `CudbTrafficControlManager` class.
- Section 2.4.13 on page 31: Added section on the `CudbTrafficBlockingRule` class.
- Section 2.5 on page 32: Added section on Ericsson Common Information Model (ECIM) structures.
- Section 2.6.1 on page 33: Added a constraint for `ldapRootPassword` attribute in Table 28.
- Section 2.6.1 on page 33 and Section 2.6.4 on page 37: Updated note with information on new `applyConfig` action.
- Section 2.6.4 on page 37, Section 2.6.5 on page 40, and Section 2.6.6 on page 40: Updated information on function availability.
- Section 2.9 on page 47: Added section on administrative operations.
- Section 4.2.1 on page 57 and Section 4.2.2 on page 60: Added new subsections to Section 4.2 on page 57.

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

Data objects and attributes are configured within the CUDB system. The objects form a hierarchy, where each object is identified by a unique Relative Distinguished Name (RDN) within the hierarchy. 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 Common Operation and Maintenance (COM) model. For more information on COM, see *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.	• 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.	• Optional: attributes can be set optionally. • Mandatory: attributes have to be at object instance. • Restricted: attributes can only be set at object instance creation. • Read only: attributes can only be read. These values are obtained from the system and are read from the BC server. In case the connection to the BC server is down, their value will be set as `emptySet` returning empty square brackets: `[]`. • Read/Write: attributes can be read and written. • Write only: attributes can only be written. • Multivalued: multiple values following the format `[<value>, <value2>...]`. These values can be defined sequentially, especially when adding a new one.

Note: The `userLabel` attribute is an optional and EcimString type attribute for operator free use. This attribute is omitted from the tables describing the attributes of the different objects.

2.1 Data Types

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

Table 2 Data Types

Data Type	Description
`CudbIPAddress`	Represents an IP address (only IPv4). A version 4 IP address is specified in dotted quad notation, for example `130.100.92.154`. *RFC 791*, Reference [26] describes the syntax details.

Table 2 Data Types

Data Type	Description
EcimAction	Administrative operation.
EcimBoolean	Has two values, true or false.
EcimEnumeration	Used to enumerate a list of literals consisting of both a name and a value.
EcimPasswordString	Set of characters representing a password. Appears masked in log.
EcimString	Set of characters.
EcimStruct	Group of different data types that are scoped together.
EcimUint32	Unsigned integer 32 bits.
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	Specify time period between 1ms - 3600000 ms (1 hour).
NumericString	A string representing a number.
Time	Represents the local time as portrayed in its international standard (ISO 8601).

2.2 Class Hierarchy

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

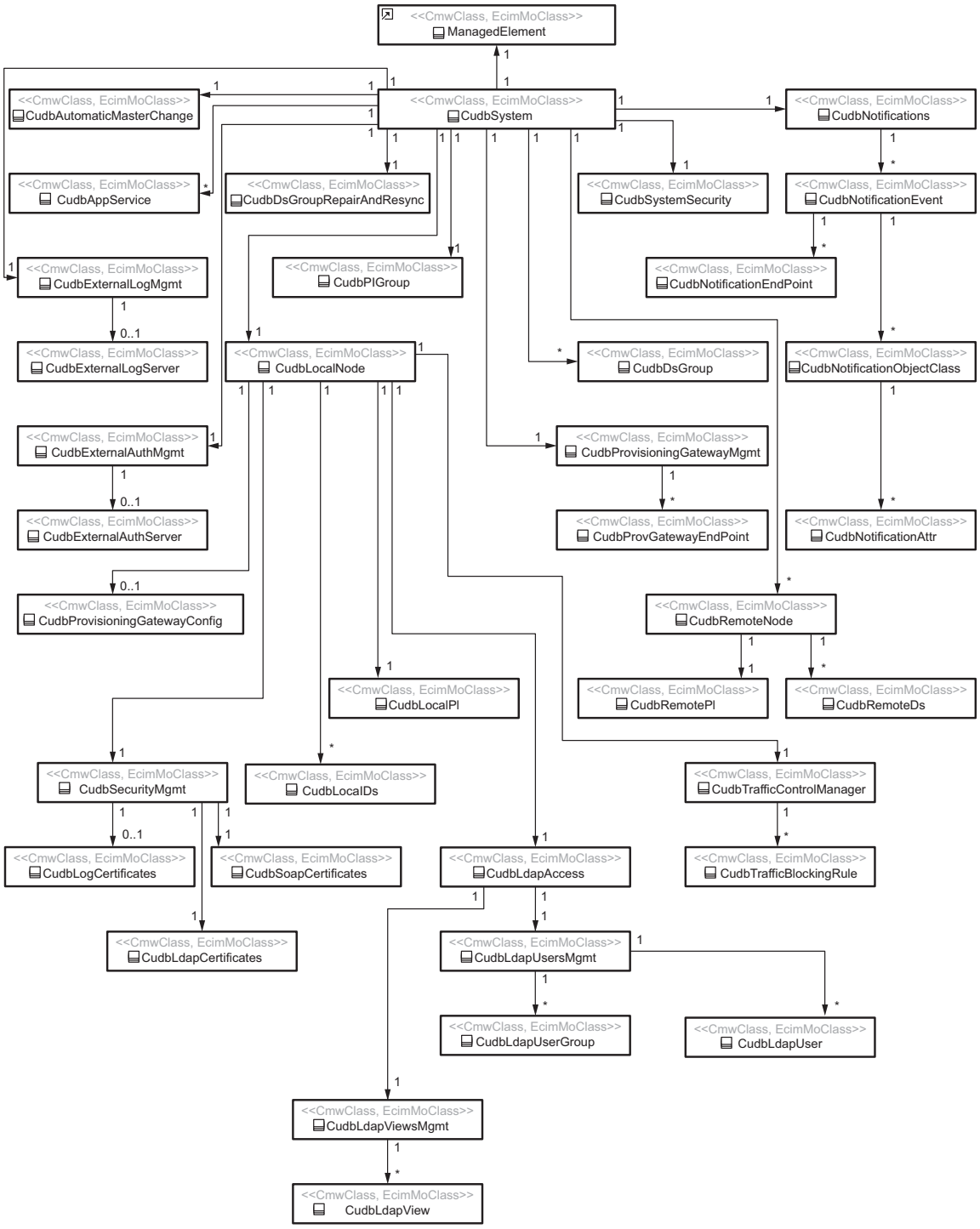

Figure 1 Class Hierarchy

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 for all CUDB Management Object Model (MOM). There is just one instance per CUDB node and it cannot be deleted in 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
`automaticServiceContinuity` Attribute for enabling/disabling the automatic triggering of Service Continuity for minority scenarios. Refer to the *Service Continuity for Asymmetrical Split Scenarios* section of *CUDB High Availability*, Reference [2] for more information.	EcimBoolean Default value: ‘‘false’’	Optional Read/Write
`backboneReliability` Deprecated: has no functional behavior.	EcimBoolean Default value: ‘‘true’’	Optional Restricted
`binlogExpireDays` Binlog files older than the value of this attribute are eligible to be purged by an age-based purge.	NumericMaxInclusive255 Range: 0 - 255 Default value: "2"	Optional Read/Write
`cudbSystemId` Identifies the instance of this class.	EcimString Range: 1 Value: ‘‘1’’	Mandatory Restricted
`defaultZone` Includes all the CUDB nodes that are not explicitly included in any other geographical zone. For more information, refer to *CUDB Multiple Geographical Areas*, Reference [1].	EcimUint32 Example: ‘‘1’’	Optional Read/Write
`dsClusterDropRatioAlarmThreshold` Defines a threshold on the DS cluster ratio, that is, the number of dropped operations 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 DS High Load alarm is raised. For more information, refer to *Storage Engine, High Load In DS*, Reference [7].	NumericMaxInclusive100 Range: 0 - 100 Default value: ‘‘5’’	Optional Read/Write

Table 3 Class CudbSystem

Attribute Name	Data Type	Properties
`ldapFrontEndDropRatioAlarmThreshold` Defines a threshold on the LDAP FE ratio, that is, the number of dropped operations 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 FE High Load alarm is raised. For more information, refer to *LDAP Front End, High Load in LDAP Processing Layer*, Reference [14].	NumericMaxInclusive100 Range: 0 - 100 Default value: “5”	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 on the PLDB ratio, that is, the number of dropped operations 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 PLDB High Load alarm is raised. For more information, refer to *Storage Engine, High Load in PLDB*, Reference [8].	NumericMaxInclusive100 Range: 0 - 100 Default value: “5”	Optional Read/Write
`provisioningAssurance` Attribute for enabling/disabling 'Provisioning assurance after CUDB mastership change' feature at any time.	boolean Constraint: At least one instance of `CudbProvGatewayEndPoint` class must exist before activating the feature. Example: “true”	Mandatory Read/Write
`reallocationBlockSize` Sets the granularity of the number of Distribution Entries (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 *CUDB Subscription Reallocation*, Reference [3] is active.	EcimUint32 Range: 1 - 10000 Example: “500”	Mandatory Read/Write
`replicationTimeDelayAlarmThreshold` Defines a threshold (in seconds) on the replication delay between a slave Processing Layer Database (PLDB) or Data Store (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. If the replication delay goes above this threshold, the “Storage Engine, Replication Delay Too High In PLDB” or “Storage Engine, Replication Delay Too High In DS” alarm is raised. If this attribute is set to "0", no alarms are raised regardless of the value of the replication delay. For more information, see *Storage Engine, Replication Delay Too High In PLDB*, Reference [6] and *Storage Engine, Replication Delay Too High In DS*, Reference [5].	EcimUint32 Default value: “0”	Optional Read/Write

2.3.2 Class CudbDsGroup

Several Data Stores (DSs) are logically grouped in a CUDB system into a DS Unit Group (DSG). The `CudbDsGroup` class represents a DSG and it is used to hold configuration parameters common to all DSs belonging to a certain DSG. There are as many instances of this class as DS groups defined in a CUDB system and these instances can be deleted in 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 accessing the cluster data through SQL. The proposed 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 has to be unique in the configuration data model. Example: for DS Group 8, 15080 = 15000 + 8 * 10	Mandatory Restricted
`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 proposed value for this port follows this rule: 15000 + (DS Group number * 10) + 1. Do not modify this parameter once the node is providing traffic service.	NumericMaxInclusive65535 Range: 0 - 65535 Constraint: It has to be unique in the configuration data model. Example: for DS Group 8, 15081 = 15000 + 8 * 10 + 1	Mandatory Restricted
`masterReplicationChannel2Port` Sets the port where the master DS of the DSG listens for replication purposes. This port corresponds to replication channel 2. The proposed value for this port follows this rule: 15000 + (DS Group number * 10) + 2. Do not modify this parameter once the node is providing traffic service.	NumericMaxInclusive65535 Range: 0 - 65535 Constraint: It has to be unique in the configuration data model. Example: for DS Group 8, 15082 = 15000 + 8 * 10 + 2	Mandatory Restricted

Table 4 Class CudbDsGroup

Attribute Name	Data Type	Properties
`memoryEligibleThreshold` Sets the percentage of occupation under which the DSG is selectable for accepting distributed data from other DSGs, when 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 a 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 one service/application Front End (FE) that provides a set of object classes and attributes. There are as many instances of this class as the number of application FEs using the CUDB system and these instances cannot be deleted in 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 the `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
`appSrvName` Name of the application FE. It is the prefix of all object classes present in the schema that this application FE provides. The above description is not valid to application FEs requiring that the object classes present in the provided schema do not have prefixes.	EcimString Example: “csps”	Mandatory Restricted

Table 5 Class CudbAppService

Attribute Name	Data Type	Properties
`cudbAppServiceId` Identifies the instance of this class. The value for this identifier is relevant as it defines the order in which Lightweight Directory Access Protocol (LDAP) schemas are loaded in the system. Services/application FEs having schemas that have dependencies with other application FEs/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 CudbLdapAccess instance is loaded before any service/application FE schema.	EcimString Example: “1”	Mandatory Restricted
`ldapAppSrvSchema` Name of the file (not including path) where the LDAP schema for this application FE is stored. For more information on LDAP schema, see *CUDB LDAP Interwork Description*, Reference [10]. This attribute is modified when a schema update is performed. For more information about schema updates, see *CUDB Node Schema Update*, Reference [11].	EcimString Example: “csps.schema”	Mandatory Read/Write
`sqlAppSrvDsSchema` Name of the file (not including path), where all internal data format in the DS for this application FE is stored. This attribute is modified when a schema update is performed. For more information about schema updates, see *CUDB Node Schema Update*, Reference [11].	EcimString Example: “csps-ds.sql”	Mandatory Read/Write
`sqlAppSrvPlSchema` Name of the file (not including path), where all internal data format in the Processing Layer Database (PLDB) for this application FE is stored. This attribute is modified when a schema update is performed. For more information about schema updates, see *CUDB Node Schema Update*, Reference [11].	EcimString Example: “csps-pl.sql”	Mandatory Read/Write

2.3.4 Class CudbPlGroup

All partition layers are logically grouped in a CUDB system into a partition layer group. The `CudbPlGroup` class represents a partition layer group, and it is used to hold configuration parameters common to all PLs belonging to a certain partition layer group. There is only one instance of this class for all the PL of the CUDB system and that instance cannot be deleted in 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 proposed value for this port is 15000. Do not modify this parameter once the node is providing traffic service.	NumericMaxInclusive65535 Range: 0 - 65535 Constraint: It should be unique in configuration data model. Example: “15000”	Mandatory Restricted
`cudbPlGroupId` Identifies the instance of this class.	EcimString Range: 1 Example: “1”	Mandatory Restricted
`masterReplicationChannel1Port` Sets the port where the master PL of the PL group listens for replication purposes. This port corresponds to replication channel 1. The proposed value for this port is 15001. Do not modify this parameter once the node is providing traffic service.	NumericMaxInclusive65535 Range: 0 - 65535 Constraint: It should be unique in configuration data model. Example: “15001”	Mandatory Restricted
`masterReplicationChannel2Port` Sets the port where the master PL of the PL group listens for replication purposes. This port corresponds to replication channel 2. The proposed value for this port is 15002. Do not modify this parameter once the node is providing traffic service.	NumericMaxInclusive65535 Range: 0 - 65535 Constraint: It should be unique in configuration data model. Example: “15002”	Mandatory Restricted
`memoryWarningThreshold` Sets the percentage of occupation at which the *Storage Engine, Memory Usage Too High In PLDB, Warning*, Reference [9] alarm is raised.	NumericMaxInclusive100 Range: 0 - 100 Example: “80”	Mandatory Read/Write

2.3.5 Class CudbSystemSecurity

The `CudbSystemSecurity` class captures 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 and that instance cannot be deleted in 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
cudbSystemSecurityId Identifies the instance of this class.	EcimString Range: 1 Example: "1"	Mandatory Restricted
lockoutPeriod The number of seconds the account is locked for after unsuccessful login attempts.	EcimUint32 Default value: "21600"	Optional Read/Write
maxNumFailedLogins Number of unsuccessful logins before user account is blocked.	EcimUint32 Default value: "5"	Optional Read/Write
minPasswordLength The minimum OAM password length.	EcimUint32 Default value: "8"	Optional Read/Write
minPasswordNonRepeat The minimum number of unique passwords before a password can be repeated.	EcimUint32 Default value: "12"	Optional Read/Write
secureLdapProxy Indicates that the LDAP client initiates proxy connection using Transport Layer Security (TLS).	EcimBoolean Default value: "false"	Optional Read/Write
secureMySqlReplication Indicates that the database cluster client initiates connection between nodes using TLS.	EcimBoolean Default value: "false"	Optional Read/Write
systemMonitorSafeMode Deprecated: Has no functional behavior.	EcimBoolean Default value: "true"	Optional Read/Write
tlsCaCertificatesFile 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.	EcimString 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, and it cannot be deleted.

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

Table 8 Class CudbAutomaticMasterChange

Attribute Name	Data Type	Properties
`enabled` Indicates if Automatic Mastership Change is enabled or not.	EcimBoolean Default value: "false"	Optional Read/Write
`maxReplicationTimeDelay` Defines a threshold in milliseconds on the replication delay between a slave PLDB or slave DSG cluster replica and the master PLDB or master DSG cluster replica it 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 `timeWindowEnd`) is used to define a daily time interval in which the AMC process is allowed to run. Related to `timeWindowEnd`, the attribute can be configured as follows: • If `timeWindowStart` = `timeWindowEnd`, then AMC is always enabled. • If `timeWindowStart` < `timeWindowEnd`, then AMC is enabled in the defined time period every day. • If `timeWindowStart` > `timeWindowEnd`, then AMC will be enabled from the time defined with `timeWindowStart` until the time defined with `timeWindowEnd` on the next day.	Time Default value: "00:00:00"	Optional Read/Write
`timeWindowEnd` This attribute (along with `timeWindowStart`) is used to define a daily time interval, in which the AMC process is allowed to run. See the above row of`timeWindowStart` 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 hold 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, and it 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
`automaticBackupRestoreEnabled` Indicates whether Self-Ordered Backup and Restore is enabled or not.	EcimBoolean Default value: "true"	Optional Read/Write

Table 9 Class CudbDsGroupRepairAndResync

Attribute Name	Data Type	Properties
autoSRCCAndDREnabled Indicates whether automatic execution of Selective Replica Check and Data Repair is enabled or not.	EcimBoolean Default value: "true"	Optional Read/Write
cudbDsGroupRepairAndResyncId Identifies the instance of this class.	EcimString 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
CudbExternalAuthMgmt Identifies the instance of this class.	EcimString Range: 1 Example: "1"	Mandatory Restricted
enabled Indicates if the feature is activated or not.	EcimBoolean Default value: "false"	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,Cudb
ExternalAuthServer=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	CudbIPAddress Note: Strictly IP address syntax, host names are not allowed. Example: "10.1.5.15"	Mandatory Read/Write
`secondarySever` IP of the secondary external authentication server	CudbIPAddress Note: Strictly IP address syntax, host names are not allowed. Example: "10.1.5.15"	Optional Read/Write
`tlsEnabled` Indicates if TLS will be used in the connection with the external server. If TLS is enabled, the certification authority certificate has to be part of the file introduced in the `tlsCaCertificatesFile` attribute of the `CudbSystemSecurity` class.	EcimBoolean Default value: "false"	Optional Read/Write

2.3.10 Class CudbExternalLogMgmt

The `CudbExternalLogMgmt` class is used as a configuration container of 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` Indicates 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,Cud
bExternalLogServer=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.	CudbIPAddress Note: Strictly IP address syntax, host names are not allowed. Example: "10.1.5.15"	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 represents the CUDB local node. Only one instance of this class is present in each CUDB node and that instance cannot be deleted in the configuration model.

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

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<CUDB_Local_No
de_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 `applyConfig` administrative operation.(1) For more information, see Section 2.5.1 on page 32.	`EcimStruct`(2)	Mandatory Restricted
`cudbLocalNodeId` Identifies the instance of this class.	EcimString Range: Integer, not zero. Constraint: It must be different for each CUDB node in a CUDB system. Example: “1”	Mandatory Restricted
`cudbVIP` Indicates the default virtual IP address that other CUDB nodes have to use to exchange any kind of traffic with this local node.	EcimString Note: Strictly IP address syntax, host names are not allowed. Example: “10.1.5.15”	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, see *CUDB High Availability*, Reference [2].	EcimBoolean Default value: “true”	Optional Read/Write

Table 14 Class CudbLocalNode

Attribute Name	Data Type	Properties
`hwType` The type of hardware used in the node.	EcimString Allowed values: • "EBS_GEP3" • "EBS_GEP5" • "vCUDB_2CPU_6GB" • "vCUDB_16CPU_47GB" Example: "EBS_GEP5"	Mandatory Restricted
`networkElementName` Network Element Name required for this CUDB node. It is the prefix for XML counter file names (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"	Mandatory Read/Write
`oamVIP` Indicates the virtual IP address that any external application has to use to exchange OAM related traffic with this local node. It is usually referred as OAM VIP.	EcimString Note: Strictly IP address syntax, host names are not allowed. Example: "10.1.5.14"	Mandatory Restricted
`siteId` This is the site where the node is located. For more information, see *CUDB High Availability*, Reference [2].	EcimUint32 Example: "1"	Mandatory Restricted
`systemMonitorKey` Deprecated: Has no functional behavior.	EcimString Default value: ""	Optional Read/Write
`trafficVIP` Indicates the virtual IP address that any external application has to use to exchange LDAP traffic with this local node. It is usually referred as CUDB_FE VIP.	EcimString Note: Strictly IP address syntax, host names are not allowed. Example: "10.1.5.13"	Mandatory Restricted
`updateUserInfoStatus` Shows the status of an asynchronous `updateUserInfo` administrative operation.(1) For more information, see Section 2.5.1 on page 32.	EcimString(2)	Mandatory Restricted
`zone` This is the zone to which the node belongs. For more information, see *CUDB Multiple Geographical Areas*, Reference [1].	EcimUint32 Default value: "0"	Optional Restricted

(1) The `progressPercentage` attribute of the structure is never updated.

(2) The attribute value points to a certain instance of the `CudbAsyncActionProgress` structure.

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` may be present. At least one instance of `CudbLocalPl`/`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` Specifies if this 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
`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: • 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

Table 15 Class CudbLocalPl

Attribute Name	Data Type	Properties
`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. These instances can be deleted in 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. Indicates 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 has to 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 10	NumericMinInclusive1 Range: 1 - 255 Constraint: It must correspond to an existing instance of `CudbDsGroup`. Example: "3"	Mandatory Restricted

Table 16 Class CudbLocalDs

Attribute Name	Data Type	Properties
`enabled` Specifies if this 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: • 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 holding 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 and that instance can be deleted in 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
`cudbProvisioningGatewayConfigId` Identifies the instance of this class.	EcimString Range: 1 Value: “1”	Mandatory Restricted
`pgNodeIpAddresses` 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.	EcimString Syntax: <ip1>[:JMXport1:JNDIport1][;<ipn>:JMXportn:JNDIportn] where "n" is each IP address for the specific PG node. Separator for IP addresses within a node: semicolon (";") Example: One node with two IP addresses “:[10.1.33.141:9994:4099 ;10.1.33.142:8994:8099] ” 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]”	Mandatory Read/Write multivalued
`pgUserName` The user name of the PG nodes.	EcimString Example: “pgUser”	Mandatory Read/Write
`pgUserPassword` The user password of the PG nodes. It is not stored in plain text.	EcimPasswordString Example: “0pgUser1Pwd”	Mandatory Write only

2.4.5 Class CudbSecurityMgmt

The `CudbSecurityMgmt` class is used as a container holding the classes used to specify the configuration of security mechanisms based on TLS. Only one instance of this class is present in each CUDB node and that instance cannot be deleted in 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 and that instance cannot be deleted in 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 is used to specify the configuration of the Hypertext Transfer Protocol Secure (HTTPS/TLS) to secure communications using 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 and that instance cannot be deleted in 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
`cudbSoapCertificatesId` Identifies the instance of this class.	EcimString Range: 1 Value: “1”	Mandatory Restricted
`tlsCertificateFile` The path and file name of the file containing SOAP client certificate.	EcimString Default value: "" Example: “/cluster/certificates/soap/servercert.pem”	Optional Read/Write
`tlsCertificateKeyFile` The path and file name of the file containing the private key that matches the certificate stored in `tlsCertificatesFile`.	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 provided that 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

Atrribute Name	Data Type	Properties
`cudbRemoteNodeId` Identifies the instance of this class. It has to be different for each CUDB node in a CUDB system. It is the node identification of the remote CUDB node.	NumericString Range: 1-255. Example: “5”	Mandatory Restricted
`cudbVIP` Indicates the default virtual IP address the local node uses to exchange any kind of traffic with the remote node represented by this class instance.	CudbIPAddress Note: Strictly IP address syntax, host names are not allowed. Example: “10.1.5.15”	Mandatory Restricted
`enabled` When set to false , it disables the CUDB remote node without deleting this instance object, that is, a non-existing node. For more information, see *CUDB High Availability*, Reference [2].	EcimBoolean Default value: “true”	Optional Read/Write
`hwType` The type of hardware used in the node.	EcimString Allowed values: • “EBS_GEP3” • “EBS_GEP5” • “vCUDB_2CPU_6GB” • “vCUDB_16CPU_47GB” Example: “EBS_GEP5”	Mandatory Read/Write
`oamVIP` Indicates 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 has to be the same as the one stated in attribute `cudbVIP` in this `CudbRemoteNode` 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.	CudbIPAddress Note: Strictly IP address syntax, host names are not allowed. Example (default configuration): “10.1.5.15”	Mandatory Restricted
`siteId` This is the site where the node is located. For more information, see *CUDB High Availability*, Reference [2].	EcimUint32 Example: “1”	Mandatory Restricted
`systemMonitorKey` Deprecated: Has no functional behavior.	EcimString Default value: “”	Optional Read/Write multivalued
`trafficVIP` Indicates 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 has to be the same as the one stated in attribute `cudbVIP` in this `CudbRemoteNode` 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.	CudbIPAddress Note: Strictly IP address syntax, host names are not allowed. Example (default configuration): “10.1.5.15”	Mandatory Restricted
`zone` This is the zone to which the node belongs. For more information, see *CUDB High Availability*, 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 may be present (and must be created in the same commit than its parent). At least one instance of CudbLocalPl/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
enabled Specifies 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 whole CUDB system.	Mandatory Restricted
instanceState State of the remote PLDB replica. Following are the possible values: • 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. These instances can be deleted in 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. Indicates 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 has to 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 10.	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

Table 23 Class CudbRemoteDs

Attribute Name	Data Type	Properties
`instanceState` State of the remote DS replica. Following are the possible values: • 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 specifies 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
`cudbLogCertificatesId` Identifies the instance of this class.	EcimString Range: 1 Value: “1”	Mandatory Restricted
`tlsCertificateFile` The path and name of the file containing the client certificate used for TLS communication by the Centralized Security Event Logging function.	EcimString Example: `/cluster/certificates/log/cert.pem`	Mandatory Read/Write
`tlsCertificateKeyFile` The path and name of the file containing the private key that matches the certificate stored in `tlsCertificatesFile`.	EcimString Example: `/cluster/keys/log/key.pem`	Mandatory Read/Write
`logServerName` The name of the server, it has to be the same name used in the server certificate generation.	EcimString 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 and that instance cannot be deleted in 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
`cudbTrafficControlManagerId` Identifies the instance of this class.	EcimString Range: 1 Value: "1"	Mandatory Restricted
`adminState` Defines the administrative state of the function.	BasicAdmState Available values: • **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: • **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. These instances can be deleted in 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
`cudbTrafficBlockingRuleId` Identifies the instance of this class.	EcimString Range: Integer, not zero. Example: “1”	Mandatory Restricted
`blockedVIP` IP address which will be blocked.	EcimString Note: Strictly IP address syntax, host names are not allowed. Example: “10.1.5.15”	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 hold 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
`id` Identifies the instance of this class.	EcimString Range: 1 Example: “0”	Mandatory Restricted
`actionId` Uniquely identifies the invocation of an action.	EcimUint32 Example: “0”	Read only
`actionName` Name of the invoked asynchronous action.	EcimString Example: `applyConfig`	Read only

Attribute Name	Data Type	Properties
additionalInfo Used for logging significant information.	EcimString Example: applyConfig automatically makes any configuration model change persistent.	Read only
progressInfo Textual information that describes the current state of the action execution.	EcimString Example: applyConfig execution running.	Read only
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/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 struct member state 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 is used to specify CUDB-related LDAP access configuration in the CUDB local node “holding” it. There is just one instance per CUDB node and it cannot be deleted in 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
`cudbLdapAccessId` Identifies the instance of this class.	EcimString Range: 1 Value: 1	Mandatory Restricted
`cudbRootEntryDn` 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.	EcimString 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: • If the type of a `rootDN` naming attribute is defined as case insensitive (has `caseIgnoreMatch` value in the `EQUALITY` property of the attribute definition in the LDAP schema), its value must be specified in lowercase. • If the type of a `rootDN` 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: • “dc=telco-op,dc=com” /** `dc` is defined with `EQUALITY caseIgnoreMatch` in the schema) **/ • “attr=Telco_OP,dc=com” /** `attr` is defined as case sensitive in the schema and “Telco_OP” is the value passed in the LDAP operations to read or write entries **/	Mandatory Restricted

Table 28 Class CudbLdapAccess

Attribute Name	Data Type	Properties
`customDistributionPolicyEnabled` Indicates if a custom distribution policy library is loaded. When this attribute is changed to "true", the library is loaded and when changed to "false", the library is unloaded. For more information about distribution algorithms, see *CUDB LDAP Data Access*, Reference [4].	EcimBoolean Default value: "false"	Optional Read/Write
`ldapAttrIndexes` List of LDAP attributes (defined in some of the LDAP schemes managed in CUDB system-LDAP Access level) to be managed as "searching indexes". For more information, see *CUDB Application Integration Guide*, Reference [13].	EcimString Syntax: LDAP attribute Constraint: When modifying this attribute, it is only allowed to add indexes, but not to remove any existing index. `LdapAttrIndexes` must be defined in the same order on every node. Example: "MSISDN","IMSI" International Mobile Subscriber Identity (IMSI). Mobile Station ISDN Number (MSISDN).	Optional Read/Write multivalued
`ldapRootPassword` Password of the LDAP `rootdn` user.	EcimString Default value: "**********" Constraint: This attribute can contain only ASCII alphabetic characters, numeric digit characters, and the following symbols: **,-%=?+~_** For more information, refer to *CUDB Users and Passwords*, Reference [17].	Optional Read/Write
`nodeLdapAuth` 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. (1) For more information, see *CUDB Security and Privacy Management*, Reference [12].	EcimString Default value: "SIMPLE"	Optional Read/Write

Table 28 Class CudbLdapAccess

Attribute Name	Data Type	Properties
`nodeLdapHash` Indicates 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. For more information, see *CUDB Security and Privacy Management*, Reference [12].	EcimString Default value: "SHA-256"	Optional Read/Write
`redundancyLevel` 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, see *CUDB High Availability*, Reference [2].	NumericMaxInclusive255 Example: "4"	Mandatory Read/Write

(1) If the value is "SIMPLE" then the LDAP user will not be able to use Simple Authentication and Security Layer (SASL) authentication. 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 in clear text.

Note: The value of the `ldapRootPassword` attribute can be retrieved right after it has been set, if the `applyConfig` action has not yet been executed. Nevertheless, once the `applyConfig` administrative operation is executed, the value shown for the newly set password is "*********".

2.6.2 Class CudbLdapUsersMgmt

The `CudbLdapUsersMgmt` class is used as a container of CUDB LDAP users and CUDB LDAP users groups. There is just one instance per CUDB node and it cannot be deleted in 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
`cudbLdapUsersMgmtId` Identifies the instance of this class.	EcimString 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 in 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
`cudbLdapUserGroupId` Identifies an LDAP user group used to compose the LDAP Organizational Unit (OU) used as parent for LDAP users.	EcimString 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 in 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, see *CUDB LDAP Data Access*, 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. (1)	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, see *CUDB Security and Privacy Management*, Reference [12]. Example: “admin1”	Mandatory Restricted

Table 31 Class CudbLdapUser

Attribute Name	Data Type	Properties
`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, see *CUDB Security and Privacy Management*, Reference [12]. Example: “”	Mandatory Restricted
`cudbUserPassword` The password of the LDAP user. For more information, see *CUDB Security and Privacy Management*, Reference [12].	EcimPasswordString Constraint: `cudbUserPassword` cannot be an empty string. Example: “0admin1Pwd”	Mandatory Read/Write
`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, see *CUDB High Availability*, Reference [2].	EcimBoolean Default value:“ false”	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 *CUDB LDAP Data Access*, 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: • 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. 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.	EcimString	Mandatory Read/Write

Table 31 Class CudbLdapUser

Attribute Name	Data Type	Properties
`readModeInPL` Used to determine which PLDB replica is used for read LDAP requests when an access to PLDB is required. Following are the possible values: • 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. See the end of this table for information on the supported value combinations.	EcimString	Mandatory Read/Write
`userLdapAuth` Determines if the password of the LDAP user is either stored in clear text or hashed. If the value is empty, the value of `userLdapAuth` in LDAP database is equal to `nodeLdapAuth` at the time when the user is created. 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 `userLdapHash` attribute in this ObjectClass, if present, or the `nodeLdapHash` attribute in the `CudbLdapAccess` instance. (2) For more information, see *CUDB Security and Privacy Management*, Reference [12].	EcimString	Optional Read/Write
`userLdapHash` Indicates type of hash chosen to store the password when the `userLdapAuth` attribute is set to "SIMPLE", otherwise this attribute is not applicable. If the value is empty, the value of `userLdapHash` in LDAP database is equal to `nodeLdapHash` at the time when the user is created. If the value for this attribute is set, it prevails over `nodeLdapHash` specified in `CudbLdapAccess` instance. For more information, see *CUDB Security and Privacy Management*, Reference [12].	EcimString	Optional Read/Write
`cudbLdapViewId` Identifier of `CudbLdapView` attached to the actual user. An LDAP view cannot be assigned to a provisioning or re-provisioning user.	EcimString Constraint: has to correspond with the value of the `cudbLdapViewId` of a `CudbLdapView` object.	Optional Read/Write
`isReProvisioningUser` This parameter is used by a PG user to send reprovisioning operations after a mastership change. If a user is assigned to an LDAP View, it cannot become a re-provisioning user. For more information, see *CUDB High Availability*, Reference [2].	boolean Example: "true" Default value: "false"	Optional Read/Write

(1) The prefix "internal" in the `cudbLdapUserId` value is reserved for Ericsson internal use.

(2) If the value is "SIMPLE" then the LDAP user will not be able to use SASL authentication. If the value for this attribute is set, it prevails over `nodeLdapAuth` specified in `CudbLdapAccess` class.

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 HSS and HLR).
- `readModeInPL=MA` and `readModeInDS=MA` for provisioning applications and application FEs (such as the Provisioning Gateway).

For more information, see *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 represents a container for views. Only one instance of this class is present in each CUDB node and it is created at installation and cannot be deleted.

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
`cudbLdapViewsMgmtId` Identifies the instance of this class.	EcimString 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 represents the 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
`cudbLdapViewId` Identifies the instance of this class.	EcimString Constraint: this attribute must be unique in the whole CUDB system.	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.

2.7 Notifications Object Classes

Notifications object classes provide information about notifications to external network entities, for example, Home Subscriber Server (HSS)/Subscription Locator Function (SLF) FE and Home Location Register (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 via these classes.

For more information, see *CUDB Notifications*, Reference [18].

2.7.1 Class CudbNotifications

The `CudbNotifications` class is the root element for the notifications application FE. Instance of this class cannot be deleted.

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
`cudbNotificationsId` Identifies the instance of this class.	EcimString Range: 1 Value: 1	Mandatory Restricted
`enabled` Indicates if notifications are to be sent to endpoints. If set to "true", notifications are sent. If set to "false", notifications are not sent, and monitoring of data is stopped.	EcimBoolean Default value: "true"	Optional Read/Write
`maxReattempts` The maximum number of retries to send a notification to a 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 only once to a FE.	NumericMaxInclusive255 Default value: "3"	Optional Read/Write
`reattemptTime` The base time (in milliseconds) between attempts to send a notification to a FE.	NumericRangeInclusive1to 3600000 Default value: "1000"	Optional Read/Write

2.7.2 Class CudbNotificationEvent

The `CudbNotificationEvent` class defines a notification event to be sent to an application FE when a monitored CUDB object class attribute changes its value. Instance of this class cannot be deleted in the configuration model.

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 *CUDB SOAP Interwork Description*, Reference [19].	EcimString Example: "mobilityEvent"	Optional Read/Write

2.7.3 Class CudbNotificationEndPoint

The `CudbNotificationEndPoint` class defines a notification endpoint that receives the specified notification event. Instance of this class cannot be deleted in the configuration model.

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: "https://127.0.0.1:8080"	Mandatory Read/Write
`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 defines a CUDB subscriber object class whose attributes are involved in the notification. Instance of this class cannot be deleted in the configuration model.

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 partial DN of the object/attribute to be monitored. The partial DN hangs from a DE. The values for the attributes in the DN must be in normalized form, which means that if the attribute type is case insensitive (this is specified in the LDAP schema), then the value of the attribute must be written in small 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 Example: “EpsDynInfId=EpsDynInf,EpsStaInfId=EpsStaInf,serv=eps”	Mandatory Read/Write
`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: • 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 for the case when the attribute is a single value and the object class to which the attribute belongs is added or deleted in the same LDAP operation as the one that is changing the value of the attribute. (1) • monitorAll: The data is to be monitored for change. Any change in attributes below this instance can trigger a notification without exceptions. This value can be used to monitor the creation or deletion of entries with DN equal to the one configured in this object if the attribute in the leftmost RDN of the DN is configured below this object.(1) • check: The data has to 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

(1) All instances of this class type under the same `CudbNotificationEvent` instance must have the same value for the `dn` attribute.

2.7.5 Class CudbNotificationAttr

The `CudbNotificationAttr` class specifies an attribute of the CUDB object class involved in the notification. Instance of this class cannot be deleted in the configuration model.

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 “EcimString” and “EcimUint32” are supported.	EcimString Example: “PSLOC”	Mandatory Read/Write
`send` When set to “true”, the attribute is sent in the notification. When set to “false”, the attribute is not sent in the notification.	EcimBoolean Example: “false”	Mandatory Read/Write
`value` The value to be used in the comparison with the current attribute value when the instance is below a `CudbNotificationObjectClass` instance of type “check”. The attributes must be encoded as follows: • Character strings for non-binary attributes. • Base64-encoded strings with 2040 characters maximum length for binary attributes.	EcimString Example: “5”	Optional Read/Write

2.8 Provisioning Gateway Object Classes

Provisioning Gateway object classes provide information about Provisioning Gateway Endpoints, defined for configuration, related to Provisioning Assurance feature. Provisioning Gateway 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 as a container that holds the classes used to specify the configuration of Provisioning Gateway endpoints. Only one instance of this class is present in each CUDB node and that instance cannot be deleted in the configuration model.

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

```
ManagedElement=1,CudbSystem=1,CudbProvisioningGatewayMgm
t=1
```

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

Table 39 Class CudbProvisioningGatewayMgmt

Attribute Name	Data Type	Properties
cudbProvisioningGatewayMgmtId Container class that holds the Provisioning Gateway endpoint instances.	EcimString Example: “1 ”	Mandatory Restricted

2.8.2 Class CudbProvGatewayEndPoint

The CudbProvGatewayEndPoint class defines a Provisioning Gateway (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 in 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
cudbProvGatewayEndPointId Provisioning Gateway endpoint definition	NumericString Range: 1-255 Example: “1”	Mandatory Read/Write
user User to connect with Provisioning Gateway through htpp.	EcimString Example: “pgHttp1”	Mandatory Read/Write
password Password to connect with Provisioning Gateway through htpp.	EcimPasswordString Example: “0pgHttp1Pwd”	Mandatory Read/Write
replayRequestURL Universal Resource Locator (URL) in URI format to send to the Provisioning Gateway a request to start reprovisiong.	EcimString Syntax: URI format as follows: http://<PG OAM VIP>:<replay port>/<replay URL suffix> Example: “http://10.250.2.139:8282/replayer/execute”	Mandatory Read/Write
replayStatusURL URL in URI format to ask to the Provisioning Gateway the status of the reprovisioning.	EcimString Syntax: URI format as follows: http://<PG OAM VIP>:<status port>/<status URL suffix> Example: “http://10.250.2.139:8282/replayer/state”	Mandatory Read/Write

2.9 Administrative Operations

This section describes the administrative operations available in CUDB. 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
`CudbLocalNode`	`applyConfig` Administrative operation used for activating configuration changes.	N/A	Asynchronous
`CudbLocalNode`	`updateUserInfo` 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`, holds information about the current state of the `applyConfig` execution. See Section 2.4.1 on page 18 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 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 been 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.

Note: 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 57 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 57 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>,applyConfig`

Example 2

1. Establish a NETCONF session towards the CUDB node. For more information on how to establish a NETCONF session, refer to *COM NETCONF Interface Base*, Reference [25].

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 `applyConfig` attribute.	`state=RUNNING`
	`result=NOT_AVAILABLE`
		`state=FINISHED`
		`result=SUCCESS`
		`resultInfo=Ready.`
			`state=FINISHED`
			`result=SUCCESS`
			`resultInfo=<information_about_warning>: Ready.`
				`state=FINISHED`
				`result=SUCCESS`
				`resultInfo=<information_about_error>`

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

2.9.2 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, holds information about the current state of the `updateUserInfo` execution. For more information about the `updateUserInfoStatus` attribute, see Section 2.4.1 on page 18.

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.

Note: Configuration changes should not be performed 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 logged to 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 has 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 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 is FAILURE and the resultInfo reports the successful update of LDAP users and groups, but an error with configuration file generation, contact the next level of support.

2.9.2.5 Examples of Use

See the below procedures for examples of using updateUserInfo:

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 57 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>,up
   dateUserInfo
   ```

Example 2

1. Establish a NETCONF session towards the CUDB node. For more information on how to establish a NETCONF session, refer to *COM NETCONF Interface Base*, Reference [25].

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 `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 `updateUserInfoStatus` 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

Restricted for Ericsson personnel. The `cancelApplyConfig` administrative operation is used for canceling a hanging `applyConfig` execution.

3 Initial Configuration

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

3.1 Considerations

CUDB configuration is made on a per CUDB node basis. So, when configuring complete system, 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 has to 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 neither in two classes `CudbLocalNode` in different CUDB nodes nor in two classes `CudbLocalNode` or `CudbRemoteNode` in the same CUDB node. A CUDB node has the same identifier as `CudbLocalNode` in one CUDB node and as `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 CUDB system.
- All SQL schema related attributes (`sqlAppSrvPlSchema` and `sqlAppSrvDsSchema`) must be equal in **all** CUDB nodes across CUDB system.
- When including new instances belonging to classes `CudbLocalPl` or `CudbLocalDs` in a CUDB node, instances `CudbRemotePl` and `CudbRemoteDs` (belonging to `cudbRemoteNode` instance corresponding to CUDB node where physically clusters have been created) must be added in **all** other CUDB nodes in the CUDB system including same information.

4 Configuration Modification Procedure

This section describes the procedure to modify the existing configuration of CUDB. 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 [17].

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 Managed Object Model (MOM) are accessible with the CUDB configuration CLI. For more information about CLI, refer to *CUDB System Administrator Guide*, Reference [20]. The steps of the object model modification procedure are as follows:

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 updated, proceed to Step 3.

3. Establish a CUDB configuration CLI session in the active System Controller (SC). Refer to the ‘‘Finding the Active System Controller Blade’’ section of *CUDB System Administrator Guide*, Reference [20] 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.

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 [21].

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 should have no 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 *COM Management Guide*, Reference [24] for more information about NETCONF.

2. Change or fetch the configuration using the NETCONF client. NETCONF clients must be configured to use a `commit at <close-session>` commit behavior. This means that changes are committed only after the session is closed. When using the NETCONF CLI client, the configuration is changed by sending NETCONF commands in an XML form. To check the result of the commit command, send a `close-session` message. The below examples show how to change the configuration model through NETCONF.

   - **Example 1**: To change the value of the attribute `enabled` in the class `CudbLocalDs`, execute 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>
  ]]>]]>
   <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 above XML file, `[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`.

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>
   ]]>]]>
```

If the commit is successful, the reply message must look 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>
```

However, if the commit fails, the reply message must look 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>
   ]]>]]>
```

- **Example 2**: To add a new instance of the `CudbDsGroup` class to the configuration model, execute 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>
   ]]>]]>
    <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]</masterReplicationChannel
                <masterReplicationChannel2Port>[Master_Replication_Channel2_Port_Value]</masterReplicationChannel
                <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>
   ]]>]]>
```

- **Example 3**: To delete a specific instance of the `CudbDsGroup` class, 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="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>
   ]]>]]>
```

- **Example 4**: To delete the value of the `userLabel` attribute of the `CudbSystem` class, execute 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>
   ]]>]]>
    <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>
   ]]>]]>
```

- **Example 5**: To get the values of all the attributes of the specific instance of the `CudbDsGroup` class, 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">
            <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>
                            <CudbDsGroup>
                                <cudbDsGroupId>[CUDB_Ds_Group_Id]</cudbDsGroupId>
                            </CudbDsGroup>
                        </CudbSystem>
                    </ManagedElement>
                </filter>
            </get>
        </rpc>
   ]]>]]>
```

Refer to *COM Management Guide*, Reference [24] for more information on NETCONF. Refer to *COM NETCONF Interface Base*, Reference [25] for more information on how to use 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">
            <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>
   ]]>]]>
```

Glossary

For the terms, definitions, acronyms and abbreviations used in this document, see *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 Node 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 Users and Passwords*, 3/00651-HDA 104 03/10

[18] *CUDB Notifications*

[19] *CUDB SOAP Interwork Description*

[20] *CUDB System Administrator Guide*

[21] *CUDB Node Logging Events*

[22] *CUDB Node Commands and Parameters*

[23] *CUDB Glossary of Terms and Acronyms*

Other Ericsson Documents

[24] *COM Management Guide*

[25] *COM NETCONF Interface Base*

Other Documents and Online References

[26] *IETF RFC 791 http://www.rfc-editor.org/rfc/rfc791.txt*