

# CUDB Technical Product Description

## CUDB 1

---

TECHN PRODUCT DESCR

**Copyright**

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

**Disclaimer**

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

**Trademark List**

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



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Scope	1
1.2	Revision Information	1
1.3	Target Groups	2
1.4	Prerequisites	2
<b>2</b>	<b>Overview</b>	<b>3</b>
2.1	Supported Hardware	4
<b>3</b>	<b>Functions</b>	<b>7</b>
3.1	General	7
3.2	CUDB Functionality	7
3.3	Security	22
<b>4</b>	<b>Interfaces</b>	<b>25</b>
4.1	Reference Model	25
4.2	Protocols	25
<b>5</b>	<b>Architecture</b>	<b>29</b>
5.1	Logical Components	29
<b>6</b>	<b>Operation and Maintenance</b>	<b>31</b>
6.1	Fault Management	31
6.2	Performance Management	31
6.3	Log Management	31
6.4	Configuration and Software Management	32
<b>7</b>	<b>Configuration</b>	<b>33</b>
7.1	CUDB Node Configuration	33
7.2	CUDB System Configuration	34
<b>8</b>	<b>Characteristics</b>	<b>37</b>
8.1	Availability and Reliability	37
8.2	Scalability	37
	<b>Glossary</b>	<b>39</b>
	<b>Reference List</b>	<b>41</b>





# 1 Introduction

This document provides an overview of the Ericsson Centralized User Database (CUDB).

CUDB is an extensible, high-performance, subscriber-centric database system, able to consolidate subscriber data for several application Front Ends (FEs). This telecom-grade, real-time, geographically distributed database system is able to store both static and dynamic data, as well as subscriber-related service data.

CUDB provides an enhanced, controlled access to subscriptions and allows traffic business growth by offering a single point of provisioning, along with the effortless introduction of new services in a network through simplified subscriber and service management with significant cost savings.

---

---

## Attention!

This document applies to the node main release and all its consecutive SW drops. The delivery dates of the specific functions may diverge. For specific details on General Availability dates please check the UDM Roadmap.

---

---

## 1.1 Scope

The purpose of this document is to describe the architecture, functions, interfaces, element management, subscriber data management, and characteristics of the CUDB.

## 1.2 Revision Information

<b>Rev. A</b>	Initial release.
<b>Rev. B</b>	Product renamed to CUDB 1. Editorial changes. Revision valid for PRG/GA.
<b>Rev. C</b>	Other than editorial changes, this document has been revised as follows: <ul style="list-style-type: none"><li>• Updated virtualization terminology all throughout the document.</li></ul>



**Rev. D**

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

- Local Reads function added.
- Updated virtualization terminology throughout the document.

## 1.3 Target Groups

This document is intended as an introduction to the CUDB for network operators, network and service planners, as well as system engineers and administrators.

## 1.4 Prerequisites

Users of this document must possess a basic knowledge of data communication and telecommunication.



## 2 Overview

Ericsson evolved its telecom subscriber database used by its mobile circuit switch and packet switch networks, and its IP Multimedia System (IMS) and Evolved Packet Core (EPC) solutions. The result of the development is the Ericsson User Data Consolidation (UDC), a system based on a layered architecture and the physical-logical decoupling of the subscriber data, the application service, and the business logic into different network elements.

These elements are as follows:

- The Back End (BE), acting as the centralized database (such as CUDB).
- The Front End (FE), running the application and business logic (such as HLR-FE, HSS-FE, AAA-FE, and so on).

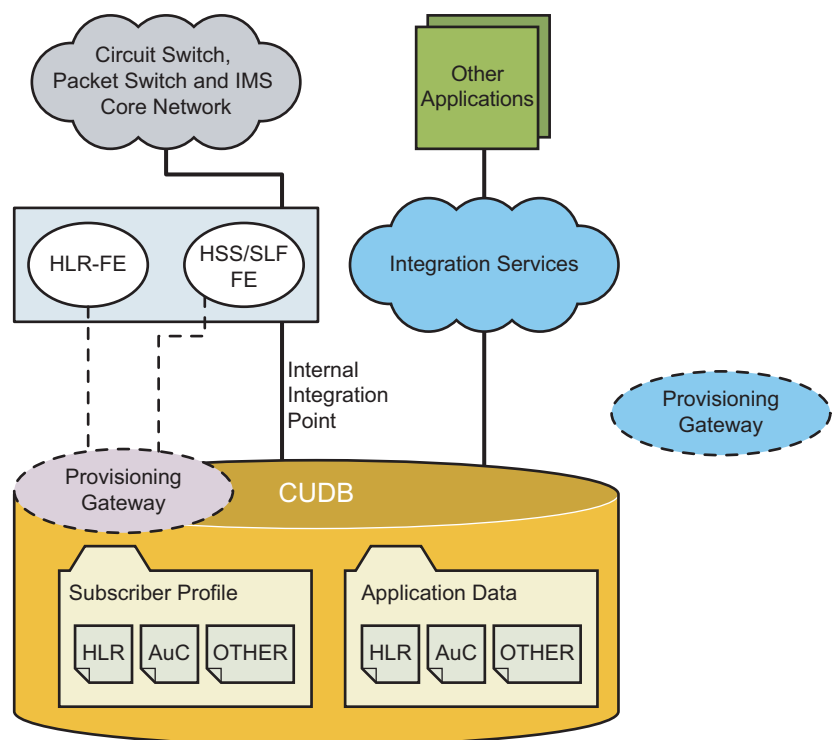


Figure 1 User Data Consolidation (UDC)

All subscriber-related data is stored and managed in the Data Layer of CUDB, while the logic of the services triggered and executed in the classic HLR monolithic architecture is performed on the HLR-FE node. Similarly, the logic of the services provided by the classic HSS/SLF Monolithic Architecture is

executed on the HSS/SLF FE node, while the subscriber data is stored in CUDB. The same strategy is applied to other application FEs.

CUDB is a distributed system of connected CUDB nodes that cooperate to provide seamless database service. A CUDB node is the main architectural unit of this network, comprising both hardware and software components. It is presented as a physical cabinet or a Virtualized Network Function (VNF) that provides access to CUDB system services. CUDB nodes can be placed in different network locations for distribution and redundancy purposes.

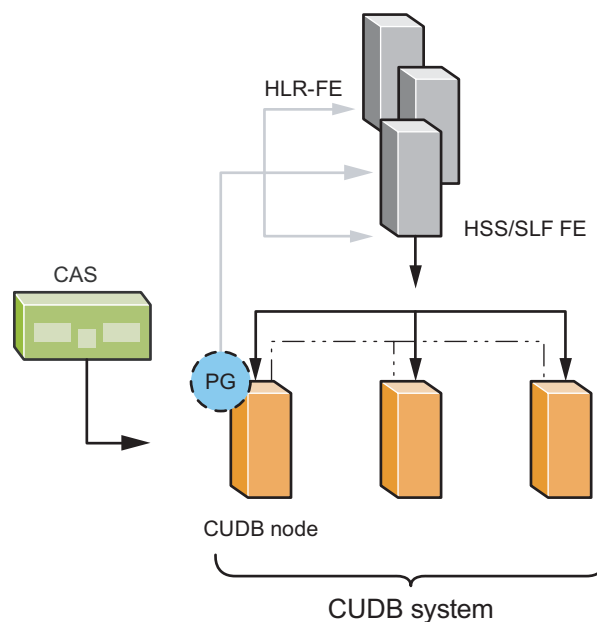


Figure 2 Ericsson User Data Consolidation Overview

CUDB stores subscriber profiles and provisioned (static) or non-provisioned (dynamic) service data associated to subscribers. As a subscriber-centric database, CUDB also holds the service profiles for the supported applications.

CUDB supports application FEs, such as HLR and HSS/SLF. Other application FEs can also be supported through integration services.

## 2.1 Supported Hardware

CUDB supports the Ericsson BSP 8100 HW platform with Generic Ericsson Processor version 5 (GEP5) blades or Generic Ericsson Processor version 3 (GEP3) blades.

It is possible to combine nodes with different type of blades (GEP3 or GEP5) in the same CUDB system. That is, hybrid systems mixing GEP3 nodes and GEP5 nodes are allowed.



**Note:** Combining GEP3 and GEP5 blades in a single cabinet (that is, in the same node) is not supported.

CUDB supports virtualized environments. CUDB is verified to run on Ericsson's Cloud Execution Environment (CEE), on BSP 8100 HW with GEP5 blades. Support for other virtualized environments or hardware is possible, but it must be secured through an integration project.





## 3 Functions

This section provides details on the functions offered by CUDB.

### 3.1 General

CUDB is a highly available, geographically redundant, real-time, distributed database exposed as a Lightweight Directory Access Protocol (LDAP) directory to HLR, AuC, HSS/SLF, or any other application.

### 3.2 CUDB Functionality

CUDB is a geographically distributed system that provides local single logical points of access for traffic and provisioning. Each CUDB node provides access to the whole subscriber base, regardless of the data distribution across the interconnected CUDB nodes. This is called a single logical point of access, physically implemented as per node IP address and port pair. The internal CUDB network and node architecture are transparent to applications.

When subscriber data is being provisioned, CUDB automatically balances the amount of stored data among the available data partitions, although custom data distribution policies can also be defined. In both cases, CUDB allows to store data for specific subscribers in CUDB nodes located in particular geographical areas.

#### 3.2.1 Structured, Flexible/Extensible Common Data Model

CUDB stores subscriber information using an LDAP Directory Information Tree (DIT) as the data model used by applications. This LDAP tree is broken into different parts managed and handled in different database clusters or cluster groups becoming a distributed LDAP directory. For further details on LDAP, refer to *CUDB LDAP Interwork Description*, Reference [1].

Subscriber data for multiple applications or network elements (that is, User Data Consolidation) is supported simultaneously.

CUDB supports the Ericsson HLR-FE, HSS/SLF FE, AuC, SAPC, M2M, AAA, MNP, ENUM, and EIR profiles.

The structured data model is flexible and extensible. This extensibility is achieved through configuration. The extension of the data model is carried out online without system degradation: new types of subscriber application profiles, application common data profiles and subscriber common data profiles can be created. In this way new services can be introduced with a shorter Time to Market (TTM) and without impact onto ongoing traffic.

CUDB supports multiple identifiers per subscriber, and new types of identifiers can also be added dynamically without system degradation.

### 3.2.2 Data Distribution

The whole subscriber data set is divided into disjoint subsets. Each of these subsets is stored in a data partition, or Data Store Unit Group (DSG). A DSG is a subscriber data partition, the basic distribution unit of subscriber data. CUDB stores every information related to a subscriber in one DSG only, but data related to different subscribers can be allocated to different DSGs.

Each CUDB node hosts a number of database clusters. These database clusters are replicated and geographically distributed across different network sites, making up a distributed Data Store Layer. These database clusters are called Data Store (DS) units.

A DS unit can be configured to host data for any DSG. For example, in Figure 3, DS2 hosts data for DSG2 in CUDB node 1 but the same DSG2 is hosted by DS1 in CUDB node 5. Not all DSGs must be hosted in all CUDB nodes. However, the data stored in all the DSGs is accessible from any CUDB node in the system.

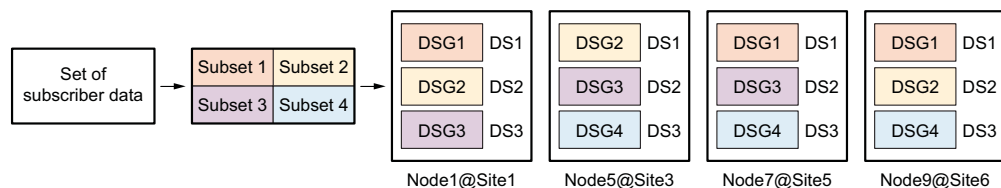


Figure 3 Subscriber Data Distribution

One of the DS units of the DSG is called the DS Master replica. All high-priority read and write operations related to the subscriber data in that partition are executed in that replica. This mechanism maintains the data consistency in the CUDB system.

The modifications are replicated asynchronously to the rest of the DS units in the partition. These other DS units are in charge of low priority tasks such as massive queries, backup and restore, or export operations. If the DS Master replica in a DSG fails for any reason, another DS unit in that partition is appointed as DS Master.

The rest of the data is placed in a specific partition called Processing Layer Database (PLDB) as aliases, profiles, and so on. This partition also stores the information about which DSG contains the data for each subscriber.

By default, subscribers are automatically distributed among the DSGs assigned to them, based on the level of occupancy of each DSG. However, it is also possible to specify the DSG (or geographical zone) where the data is stored.



CUIDB also supports custom DSG distribution algorithms during provisioning. This is performed by coding the desired algorithm rules in a dynamic library, and then loading it.

Refer to *CUIDB LDAP Data Access*, Reference [2] for further information on data distribution.

### 3.2.3 Data Backup and Restore

A backup of the whole system can be ordered and executed without stopping traffic. The backup is carried out over each subscriber subset, including all database contents. During backup, traffic is allowed, and it is not needed to stop provisioning. The resources used for this purpose are not used for traffic or provisioning.

Restore can be executed for the whole system or only for a database portion. For more details on the backup and restore procedures, refer to *CUIDB Backup and Restore Procedures*, Reference [3].

### 3.2.4 Import and Export

CUIDB supports large-scale import and export operations either for the entire database, or its selected portions. The operations are performed with the provided import and export tool commands.

Import operations can be optimized by using separate import files for different sets of data, divided by subscribers and applications, for example. In this way, a large amount of data can be imported simultaneously by using multiple connections, minimizing time.

Export can be optimized by parallel execution of different partial exports, based on filtered criteria or hierarchy/branch criteria.

For more details on data import and export, refer to *CUIDB Import and Export Procedures*, Reference [4].

### 3.2.5 Access Control

Access control policies for LDAP provisioning and traffic can be dynamically configured through the definition of users, groups, and their associated set of data access privileges.

Access Control is performed whenever an application FE attempts to access subscriber data stored in CUIDB, by means of data consumer authentication and data access authorization.

These privileges are set by using flexible Access Control Lists (ACLs) supporting regular expressions.

Control is performed depending on the following factors:

- The LDAP client type.
- The level of the service data attempted to be accessed (branch level, object class level, or attribute level).
- The type of access (read, write).

### 3.2.6 Notifications

CUDB supports outbound notifications, so whenever a piece of data is modified in a subscriber profile, CUDB can send Simple Object Access Protocol (SOAP) based notifications towards a configurable endpoint destination.

The support for notifications allows extensive configuration, including the following options:

- Monitored attribute.
- Condition sets of the monitored attribute, other attributes, or both.
- Notification message contents.
- The list of destination endpoints (application FEs).

Several receiver endpoints (that is, notification destinations) can be configured for every type of notification that has been defined. Based on the number of destinations, two types of notifications are available:

- Notifications sent to all members of the endpoint.
- Notifications sent to one member of the endpoint.

Furthermore, CUDB enables individual SOAP-based notification connections to be protected with Transport Layer Security (TLS) / Secure Socket Layer (SSL).

For more details on notifications, refer to *CUDB Notifications*, Reference [5].

### 3.2.7 EPC Mobility Support

CUDB supports HLR and HSS/SLF interoperability (based on the notification services) to notify application FEs on database updates in EPC mobility scenarios.

### 3.2.8 Geographical Zone Administration

The geographical zone concept offers tailored data distribution, allowing the reduction of bandwidth across geographically separated locations by means of



the ad-hoc distribution of subscriptions. Such distribution is based on the zone concept shown in Figure 4.

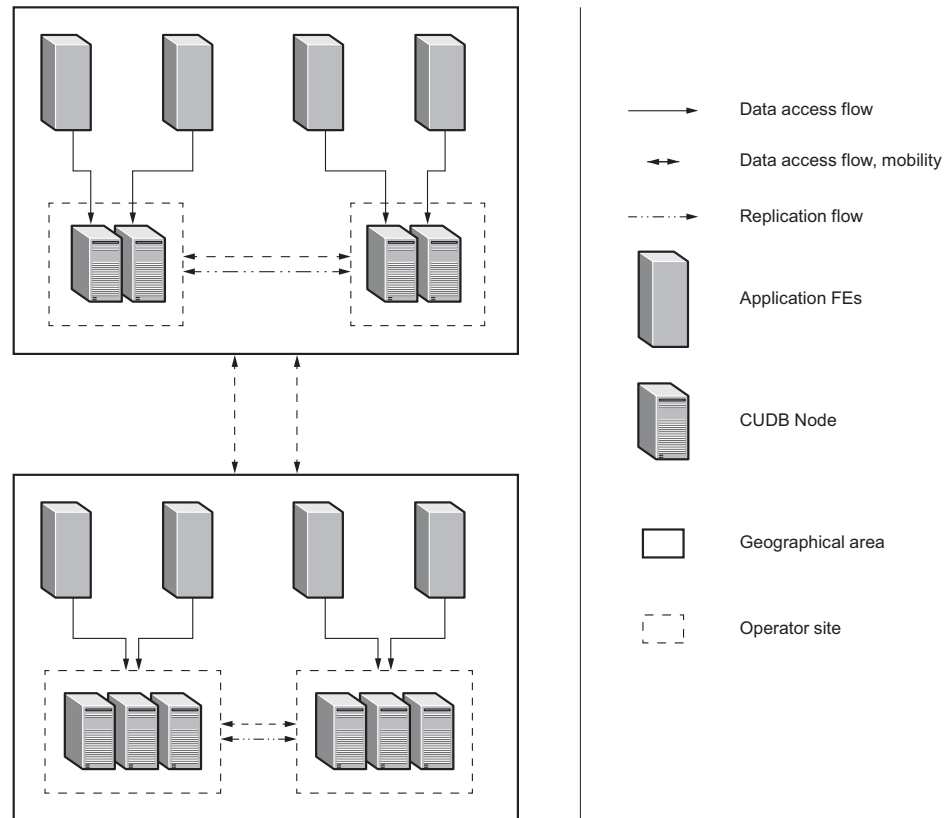


Figure 4 Geographical Zone Administration Concepts

Individual subscribers are allocated to one of the zones according to the customer's choice, reducing data flow among those areas. Each zone has its own independent storage and processing resources. When a subscriber profile is created, the subscriber is assigned to a zone. When the subscriber profile is accessed, it is generally from the zone the subscriber was allocated to, keeping the traffic in the network local. In case of mobility (for example, if the subscriber moves to another part of the country) the subscription is still transparently accessed. The rational use of zones improves data access timings and network costs for big countries, or where data transport fees are expensive, as geographical replication traffic is kept local to nodes in the zone.

For more details, refer to *CUDB Multiple Geographical Areas*, Reference [6].

### 3.2.9 BLOB Support

CUDB supports the definition of Binary Large Objects (BLOBs) in the data model, allowing to choose whether each BLOB is stored in the memory or on disk. Refer to *CUDB Binary Large Object Attributes Management*, Reference [7] for more information.



### 3.2.10 External Data Access Support

CUDB makes it possible for applications connected to CUDB to access any data stored in it (e.g. profiles for a certain service, application or both) provided that they have the proper access rights (see Section 3.2.5 on page 9).

### 3.2.11 Subscription Reallocation

CUDB supports the relocation of stored data from one DSG to other DSGs to ensure that all DSGs have an optimal level of data occupancy.

Besides DSG-level reallocation, the function also allows the node-, site-, or zone-level relocation of data by the proper handling of origin and target DSGs. The destination DSG does not need to be empty, and it does not need to belong to a new CUDB node, either.

For more details on reallocation, refer to *CUDB Subscription Reallocation*, Reference [8].

### 3.2.12 Geographically Redundant Deployments

Geographical redundancy allows storing copies of the same data in different CUDB nodes in different sites, providing a high redundancy and resiliency level. This function is available in the following two configurations:

- Double Geographical Redundancy (also known as 1+1 redundancy) deploys two geographical replicas per DSG, each one hosted in a different node located in a different site.
- Triple Geographical Redundancy (also known as 1+1+1 redundancy) deploys three replicas per DSG, each one hosted in a different node located in a different site. It is available in the Advanced Network Protection Value Package.

**Note:** Standalone configuration without geographical redundancy is supported for Customer Trial, Customer Test, and Ericsson Internal systems.

Refer to *CUDB High Availability*, Reference [9] for more information on geographical redundancy.

### 3.2.13 Layered N+1 Redundancy in HLR Classic Networks

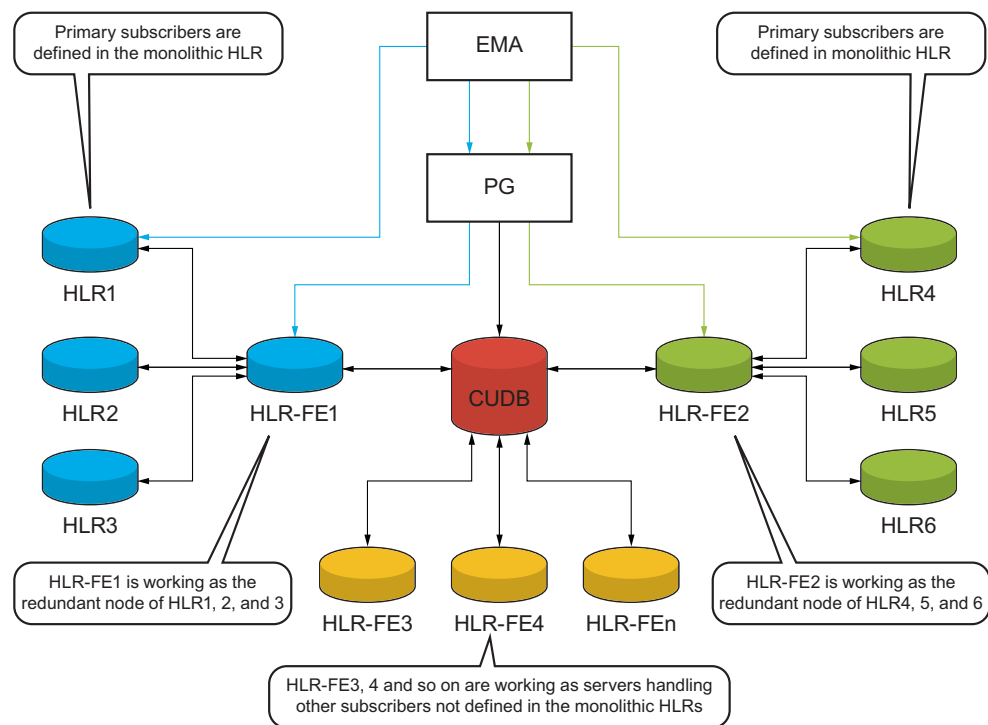
CUDB supports layered N+1 redundancy (also known as UDC hybrid solution) in HLR classic networks. The function consists of an HLR redundancy configuration, where N stands for the number of primary classic HLRs, while one is the HLR-FE standby working together with the classic HLRs.

The standard handling of N+1 redundancy is applicable in monolithic primary HLRs. In such cases, each monolithic HLR has a unique HLR-FE acting as



standby node. In such deployments, CUDB acts as the database for the HLR-FE that provides the redundancy for the non-layered classic HLRs.

The HLR-FE that works as standby node is not used in traffic under normal circumstances.



*Figure 5 Layered N+1 Redundancy Supported by a CUDB System*

Several N+1 groups can coexist in a network, each of them being handled by a specific HLR-FE connected to CUDB. In this scenario, the CUDB system used as a back end (BE) for each HLR-FE can either be the same one, or a different one for each group.

**Note:** The HLR-FE does not work in pool mechanism in this scenario.

It is possible to set other HLR-FEs attending to subscribers that have already migrated from monolithic HLRs to the data layered architecture in parallel.

### 3.2.14

#### Enhanced Overload Protection and Cooperative Load Regulation FE-BE

The CUDB nodes incorporate a self-protection mechanism, and internal load regulation between its components, so that excessive incoming traffic (overload) cannot lead to service interruption. If the extent of the overload is

up to 150% of the engineered capacity, CUDB guarantees that at least 90% of the throughput at the engineered capacity is served.

CUDB is designed to withstand traffic peaks measured up to 300% of its engineered capacity.

CUDB nodes also support the UDC Cooperative Load Regulation FE-BE mechanism that provides automated traffic throttling in the UDC system under overload. This is implemented in both the application FE and the back end (that is, CUDB), and it is triggered by the application FE when the BE is facing difficulties to handle all the incoming traffic.

In those situations, CUDB notifies the different overload situations to the application FEs in order to allow them to trigger the front end-back end Cooperative Load Regulation mechanism. Upon reception of the overload notifications, the FE can initiate a graceful throttling procedure that limits traffic throughput in the UDC system, allowing CUDB to work at its engineered capacity level, and to handle the UDC system overload situation as fast as possible as well as maximizing the service.

The front end-back end Load Regulation mechanism may occur in two different cases, as explained below:

- Overload on a DS unit in a CUDB node

When a DS gets overloaded, the CUDB node initiates its internal load regulation between the local LDAP FEs and the affected DS unit. The excess of incoming traffic to the affected DS is rejected with a specific LDAP error code.

Upon reception of this specific overload error code, the application Front End notifies to the network with the proper signalling for this case (for example, sending MAP TC-ABORT).

As the rejection is performed only for the excess of traffic towards the subscribers in the affected DS, this overload does not affect to the subscribers in other DS units, or to the UDC system as a whole.

- Overload on the PLDB or the LDAP servers in a CUDB node

In case the PLDB or LDAP servers in a CUDB node are overloaded, the excess of incoming traffic requests to the node are answered with an overload specific error code, thus indicating the node congestion situation.

Upon reception of the overload-specific error code, the application FE measures the level of congestion in the CUDB node.

Depending on the measured congestion level, the application FE may start Cooperative Load Regulation, that is, early rejection of an amount of traffic coming from the Core Network (CN) before sending it to the overloaded CUDB node.



The amount of traffic being early rejected by the application FE is dynamically adapted to the measured congestion errors received from the CUDB node.

When the overload ends, the application FE smoothly normalizes the traffic handling (that is, it smoothly decreases the traffic rejection based on the decrease of congestion errors from the BE.

Statistics on the rejected traffic under overload are logged in the affected nodes.

### 3.2.15 Traffic Prioritization per Application under Overload

The "Traffic Prioritization per Application under Overload" function makes it possible to configure how CUDB handles incoming LDAP traffic from different application FEs in scenarios where the DS layer is overloaded.

CUDB defines five levels of priority under overload: 1 to 5. LDAP users (representing application FEs) may be configured with a specific priority level through the `overloadRejectionWeight` attribute. Unless configured with a priority level, LDAP users keep the default priority level, which is 1.

In scenarios where database clusters are overloaded and traffic must be rejected, CUDB will look at the priority of the LDAP user issuing the LDAP request. Lower-priority LDAP users will see their traffic more aggressively rejected than higher-priority LDAP users. Traffic from higher-priority LDAP users is processed at the expense of traffic from lower-priority LDAP users. Thus, higher-priority LDAP users get a higher successful-to-rejected traffic ratio than lower-priority LDAP users.

It is possible to change the priority assigned to an LDAP user on the fly, even in the middle of an overload situation. Priority level changes take effect immediately.

### 3.2.16 Data Schema Management Tool

CUDB includes a Graphical User Interface (GUI) called the Data Schema Management Tool to manage the LDAP schema files. The tool allows to:

- Create new LDAP schema files.
- Browse existing LDAP schema files.
- Edit and save existing LDAP schema files.

The CUDB Data Schema Management GUI is included in the product by default. The tool only handles LDAP schema files, that is, it does not directly load the schema files from the running LDAP FEs in the CUDB nodes.

The tool allows the execution of the following operations on schema files:

- Loading several schema files within a project, as there are object classes and attributes shared between different schemas.
- Creating, opening and saving projects.
- Creating a new schema file within a project.
- Adding existing schema files to a project.
- Removing a schema file from a project.
- Updating and saving schema files.

The following actions related to LDAP schema objects can also be performed:

- Creating attribute types.
- Editing attribute types.
- Creating objects.
- Editing objects.

The tool performs integration constraints validation in the schema files loaded in a project. The checked criteria are as follows:

- Only valid and unique object identifiers (OIDs) are used (not duplicate or invalid).
- No duplicity occurs in name validation at attribute or object level.
- The schema consistency is valid (this is performed by removing objects if an attribute is used by another object class).

The Data Schema Management Tool can be executed in any Linux machine with Java and also in the Operation and Maintenance (OAM) server blades in each CUDB node.

This GUI does not directly apply any schema change to the running nodes. To proceed with the designed schema changes, Ericsson personnel need to be involved.

### **3.2.17 Consistency Check**

The CUDB Consistency Check is a basic function that looks for divergences in the data stored in a slave replica of the PLDB or a DSG, and the corresponding master replica. This function performs a consistency check by executing a scan of the database contents at both the slave side and the master side.

Consistency Checks can be ordered through the Command Line Interface (CLI), and they present the results in output log files. Alarms are raised if data



divergences are detected. The severity level of the alarm depends on the amount of divergences found.

For more details on the CUDB Consistency Check, refer to *CUDB Consistency Check*, Reference [10].

### 3.2.18 Automatic Mastership Change

The Automatic Mastership Change (AMC) function aims to eventually return the master role of PLDB or DSGs to the replica instance configured with the highest priority, in case it is now ready for service, and the master replica is elsewhere.

When AMC is enabled, it periodically checks if the PLDB and DSG masters are located on the preferred replica of each group. In case AMC detects that the PLDB master or any DSG master are not located on the preferred replica of their group, it executes a mastership movement, provided that the preferred replica is healthy and ready for service.

For more information on the AMC function, refer to CUDB High Availability.

### 3.2.19 Automatic Handling of Network Isolation

This functionality maximizes service availability for network isolation scenarios, therefore removing the potential impact of partial loss of service in an isolated site or set of sites during a network incident.

After the network isolation incident, CUDB preserves both data consistency and data durability by a sequence of automatically-triggered internal procedures, while ensuring service availability for end-users.

As a consequence of a network incident, the CUDB distributed system can be split in two or more groups of CUDB sites with no communication between each group.

A group with less than a half of the sites in the CUDB system is called a "minority". A group with more than a half of the sites is called a "majority".

One possibility is that the system is split in two halves with the same number of sites in each (symmetrical split). In this case, there are two identical halves, and no majority or minority.

The other possibility (asymmetrical split) is that there are one or more minorities, and typically one majority with half or more than half of the sites (but not necessarily).

CUDB handles these scenarios in different ways, providing different service availability per site based on the type of group each site belongs to: majority, half, or minority. The main behavior for each case and the way CUDB maximizes the service availability is described below:

- In the case of a symmetrical split situation:

The two halves of the system always provide service, what leads during the network incident to CUDB working in a mode ("multi-master" mode) where both halves execute changes but cannot replicate their local changes to the other half, thus temporarily behaving as two diverging databases.

- In the case of a asymmetrical split situation:

By default, the sites in a minority stop providing service, and only the group with the majority (or half) of the sites provides service. All the network traffic to be processed will reach this majority group via the necessary automatic Front-End and Network fail-overs. Therefore, the CUDB system suffering the network incident does not enter into "multi-master" mode and all the subsequent automatic recovery steps. No temporary diverging databases occur.

However, when needed due to a specific Network design, specific isolation situation, or any other reason, it is possible, either manually or by configuration, to force the system to provide service in a minority, thus potentially entering in "multi-master" mode too, at least for some DSGs. This allows to maximize service in the isolated network regions when the network incident and the whole network design is not allowing to redirect traffic to the majority group via automatic network fail-overs or application-FE fail-overs.

When the network connectivity is recovered, CUDB secures a proper exit of the "multi-master" mode in the cases this mode was used. This includes that all the data modifications executed during "multi-master" mode are identified, and merged together into the master (that is, the data divergences are "repaired"), in an automatic way.

In addition, as soon as the repair is finished, CUDB automatically orders the needed data backup in the current master and data restore in the current slave/slaves, in a safe and controlled way, in order to restore the alignment between master and slave replicas. This self-ordered backup and restore can be disabled by configuration, if desired, keeping it as a manual step instead.

Once restoration is finished, geographical replication is immediately restarted.

As a summary, this functionality provides the following:

- Allowing provisioning during symmetrical split situations
- Automatically entering in "Multi-master" mode in symmetrical split situations
- Allowing to enter in a "multi-master" mode (by command or configuration) in minority situations, therefore providing service also in the isolated minority when needed.
- Automatically checking the data changes missing from the master replica in force after connectivity is back, when "multi-master" mode was entered.



- Automatically repairing the data changes missing from the master replica in force, after a "multi-master" mode phase.
- Automatically ordering and executing the needed data backup in masters and data restore in slaves, as part of the recovery actions after network split.

The automatic repair is based on time stamps.

The logic in the Automatic Handling of Network Isolation function secures that provisioned data is never impacted or overwritten.

There might be some cases where the repair functionality is not able to resolve a conflict (the same LDAP entry has been modified in two sites during the network incident). In those cases, a report is generated identifying which LDAP entry needs further investigation.

### **3.2.20 Provisioning Assurance after CUDB mastership change**

This functionality preserves the provisioning data durability after an unexpected and unplanned mastership change, in order to ensure correct service availability for end-users.

After a mastership change happens, CUDB automatically addresses the potential risk of having lost, or partly executed, some provisioning operations. These operations may have not been replicated in time to the new elected master before or during the mastership fail over happened.

When this happens, CUDB informs the Provisioning Gateway to re-provision the affected operations, securing that any potentially lost (or partially executed) provisioning operation is properly recovered and not lost.

### **3.2.21 Multi-Application Support**

CUDB on BSP 8100 (GEP5) platform includes multi-application support, which allows the following:

- Co-habitation of applications within a single subrack
- Expansion within the same subrack, cabinet or with additional cabinets
- One application can span over sub-racks or even cabinets
- Common infrastructure shared by all applications
- Re-location of HW between applications via command

### **3.2.22 LDAP Data Views**

The LDAP Data Views function makes it possible for application FEs to access data stored in CUDB through a custom tree structure (that is, a custom DIT)

and with a custom schema. LDAP users can be configured to either access CUDB using the core DIT, or to use one of the defined LDAP data views.

Several LDAP data views can be defined to accommodate different kinds of application FEs.

Accessing CUDB through a data view means some restrictions to the “write” operations (add, delete, modify), in order to guarantee the coherence of the core DIT data (refer to *CUDB LDAP Data Views*, Reference [19] for more information). Provisioning-type application FEs are strongly recommended to access CUDB through the core DIT, and not through a data view.

LDAP data views are not available for export and import procedures, or for notifications. Notifications must be configured according to core DIT distinguished names and attributes, and data in SOAP messages is, likewise, identified by its core DIT distinguished names and attributes.

The LDAP Data Views function is included in the Application Facilitator Value Package.

### 3.2.23 Virtualized Network Function (VNF) support

The VNF Support function support makes it possible to decouple software and hardware through virtualization, thus enabling the harmonization of HW across multiple products and vendors, and optimizes hardware utilization.

Virtualization comes with benefits that will change the way infrastructures and applications are deployed, used, maintained and supported. It is an enabler for operators to move or deploy applications into the operator cloud.

The support of virtualization in CUDB is intended for early adopters of cloud technology.

VNF Support covers installation, configuration and adaptation of CUDB OAM functions.

Virtualized CUDB provides the same functionality and architecture of native CUDB by:

- Keeping the logical architecture. A physical blade in native is mapped to a single Virtual Machine (VM), while a cabinet is mapped to a VNF
- Supporting the same external interfaces. Interoperability with non-virtualized application front ends is guaranteed
- Mapping and adaptation of existing OAM functions. Procedures in native CUDB can also be executed in virtual deployments, with the exception of those related to hardware, networking and routing, which are handled by the applicable Virtual Infrastructure Manager.

CUDB is available through two different delivery models:



- 1 System verification delivery model: virtualized CUDB is system-verified over full Ericsson cloud infrastructure (ECEE & BSP 8100), which guarantees CUDB functionality and performance and maximizes predictability
- 2 Software-only delivery model: used when third party cloud infrastructure is required. In this case functionality and performance of the solution is secured through a system integration project and life cycle management is provided for CUDB only. In this case CUDB provides information on the minimum requirements on the cloud infrastructure.

### 3.2.24 Flexible PL Deployment

Flexible PL Deployment function allows the optimization of the number of PLDB servers in the system: operators can decide on the level of redundancy for PLDB per site, as long as at least one instance of PLDB is kept per site.

Until CUDB 1 release, a CUDB system consisted of a set of CUDB nodes of one single type: all nodes contained both a PLDB replica and some Data Store units.

With Flexible PL Deployment, it is possible to define nodes with only Data Store units and without a PLDB replica. These nodes will not be configured to receive direct traffic from application front-ends.

This function allows the optimization of equipment in existing customers by:

- 1 Reducing the number of servers required in a system
- 2 Converting redundant PLDB servers into Data Storage servers

Consequently, the allowed deployments in a CUDB system are:

- All nodes with PLDB servers (already supported in all previous releases)
- Some nodes with PLDB plus some nodes without PLDB. In this deployment nodes without PLDB will not handle direct traffic and it is mandatory to have at least one node with PLDB per site

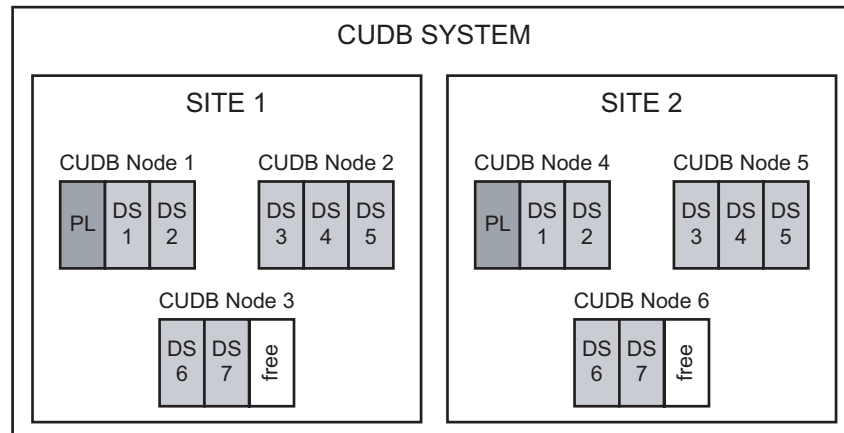


Figure 6 Example of a CUDB System setup with Flexible PL Deployment (1 PLDB instance per site)

### 3.2.25

#### Local Reads

The Local Reads function makes it possible for application frontends to read data from the closest replica, even if it is not a master replica. This means shorter response times, and reduced traffic through the IP backbone, and a more efficient use of the available processing capacity of CUDB, since the proxy traffic between CUDB nodes is minimized.

CUDB will only allow local reads on slave replicas if the data there is fresh enough, that is, if the replication distance between the slave replica and its master is not longer than a configurable threshold, defined on a per-frontend type (LDAP user) basis.

Local reads is configured per-LDAP user basis, so it is possible to have applications FEs with local reads and applications FEs without Local Reads in the same network. Local reads configuration for a given LDAP user can be changed at runtime without any traffic disturbance.

Application FEs configured to use Local Reads can choose to override the local-reads configuration and perform a read on the master replica by means of an LDAP extension control, on a per-LDAP request basis.

The Local Reads function is included in the Deployment Flexibility Value Package.

## 3.3

### Security

CUDB supports the following interfaces over a secure protocol:

- Data in CUDB can be reached through LDAP over TLS/SSL.



- Access to the CLI for configuration purposes is achieved through Secure Shell (SSH).
- For sending notifications, SOAP is used over HTTPS, with either simple or mutual authentication.
- Statistics and logs are accessible through SFTP.
- CUDB can be configured to support TLS/SSL-protected replication between nodes.

Authentication is supported by user credentials (user name and password) through the LDAP and OAM interfaces.

CUDB supports Simple Authentication, Security Layer (SASL) and TLS authentication for the LDAP interface.

Access to data can be limited by the use of ACLs. For further information on ACLs, see Section 3.2.5 on page 9.

It is possible to delegate authentication of OAM users to a properly configured, external LDAP/LDAPS authentication server.

For more details on CUDB security, refer to *CUDB Security and Privacy Management*, Reference [11].



## 4 Interfaces

This section provides information on the CUDB interfaces.

### 4.1 Reference Model

The various interfaces along with the protocols used within them are shown in Figure 7.

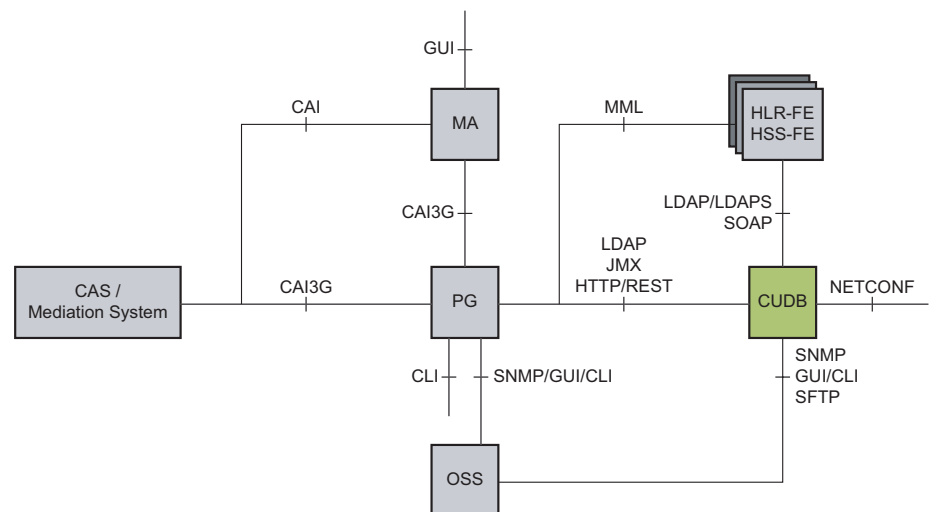


Figure 7 Reference Network Model

### 4.2 Protocols

A general outline of the protocols and the necessary stacks used by CUDB are depicted in Figure 8.

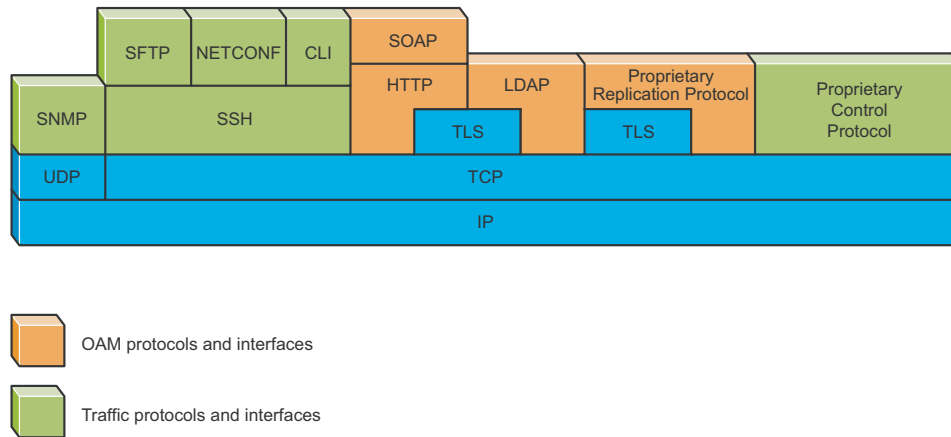


Figure 8 Protocols and Interfaces

The protocols used by CUDB and depicted in Figure 8 are as follows:

- **LDAP/LDAPS V3**

LDAP is an application protocol for querying and modifying directory services running over TCP/IP. CUDB supports the Secure LDAP protocol suite as defined by the Internet Engineering Task Force (IETF). For more information about LDAP, refer to *RFC 4510: Lightweight Directory Access Protocol (LDAP)*, Reference [21].

CUDB provides LDAP FE with the SASL Digest authentication procedure to authenticate a remote LDAP server.

CUDB supports client side authentication of the TLS/SSL LDAP interface and optional server side authentication.

- **SFTP**

The Secure File Transfer Protocol (SFTP) allows the transfer of files to and from CUDB, and is also used to retrieve performance information and log files. Data is encrypted during transfer to avoid potential packet sniffers extracting usable information from the data packets.

The protocol itself does not provide authentication and security, it relies on SSH to provide these features. Currently, there is no standard for this protocol, only IETF drafts exist.

- **SSH**

Secure Shell (SSH) is a network protocol used for data exchange through a secure channel between two computers. Encryption provides confidentiality and data integrity over an insecure network, such as the Internet. For more information about SSH, refer to



*RFC 4252: The Secure Shell (SSH) Authentication Protocol*, Reference [22].

- **CLI**

CUDB provides a Command Line Interface (CLI) for its configuration and software management. The CLI is used to alter CUDB parameters and order OAM requests, such as data backups, or software upgrade operations. Users or applications accessing the CLI need to provide login credentials.

- **NETCONF**

Besides CLI, CUDB also provides configuration management through the Network Configuration Protocol (NETCONF), from IETF. NETCONF uses XML-based data encoding and protocol. For more information about NETCONF, refer to *RFC 4741: NETCONF Configuration Protocol*, Reference [23].

- **SNMP**

The Simple Network Management Protocol (SNMP) allows logically remote users to inspect or alter the management information of a network element. CUDB supports SNMP for Fault Management (FM).

- **SOAP**

The Simple Object Access Protocol (SOAP) is a simple and extensible XML-based protocol for exchanging structured information between applications over the Internet. CUDB uses this protocol to send notifications over HTTP/HTTPS. For further details on CUDB using SOAP, refer to *CUDB SOAP Interwork Description*, Reference [12].

- **JMX**

The JMX protocol is used between CUDB nodes and PG to optionally withhold provisioning during CUDB system data backups.

- **HTTP/REST**

The HTTP/REST protocol is used between CUDB nodes and PG to trigger the provisioning assurance after CUDB mastership change function on the PG side.

CUDB also makes use of the following internal protocols:

- **Proprietary Replication Protocol**

The proprietary replication protocol is used to keep the replicas up-to-date. When the protocol is active, the changes in the database are replicated: in this way, the whole system is kept continuously updated.

- **Proprietary Control Protocol**



The proprietary control protocol is used to maintain accurate information on the status of the whole CUDB system in each node.

- **LDAP/LDAPS v3**

The LDAP protocol is also used internally between CUDB nodes for internal proxy functions.



## 5 Architecture

This section describes the CUDB architecture from two different approaches: the logical and the hardware point of view.

### 5.1 Logical Components

CUDB consists of three logical components described in the following subsections.

#### 5.1.1 Processing Layer (PL)

This component provides the following functions:

- Load balancing

CUDB offers connection load balancing by removing single points of failure and virtualizing the network.

- Processing Layer Database

The PL on a CUDB node typically holds a replicated copy of the Processing Layer Database (PLDB). The PLDB is a clustered database, holding indexing information for the data stored in the DSGs, and also storing the complete set of subscriber identities. Besides these, the PLDB can also store common subscriber data, application FE data, or any other type of data as well, if needed.

It is possible to deploy CUDB nodes without a replica of the PLDB. See the "Flexible PL deployment" function description for further information.

- LDAP Gateway and Proxy Servers

The LDAP gateway and proxy servers perform the following tasks:

- Locating subscriber information across the distributed CUDB system by looking up the subscriber identity defined in the PLDB.
- Providing protocol conversion from LDAP to the internal DS technology.
- Massive search request execution.

#### 5.1.2 Data Store (DS) Layer

This component provides the following functions:



- Data Store Unit (DS): A database cluster storing different partitions of the entire subscriber data set.
- Data Store Unit Group (DSG): A group of DS units storing exactly the same subset of subscribers (same partition) and configured in geographical redundancy replication.
- Data Replication: Replicates all modifications in each DS to the rest of DSs in the same DSG.

### 5.1.3 Monitoring and OAM Components

This component provides the following functions:

- Internal detection of the CUDB system status, failures, reacting to report alarms, service maximization, data consistency protection, and internal failover to the redundant components when needed.
- Providing capabilities for Fault Management (FM), Performance Management (PM), Configuration Management (CM), Backup and Restore, and so on.



## 6 Operation and Maintenance

CUDB provides different interfaces for different administrative purposes, such as configuration and supervision. Focusing on customer needs, the goal is to provide operators with a flexible, cost-effective and secure management system. CUDB offers Operation and Maintenance (OAM) on a per node level.

Configuration management is supported through the CLI, or the NETCONF protocol.

SNMP is used to provide alarms for Fault Management (FM). For Performance Management (PM), statistics and logs are accessible through the SSHv2 suite of tools, like SFTP.

To visualize the system status of CUDB nodes, the UDC Cockpit in User Profile Gateway (UPG) product provides a single, easy-to-use web interface.

### 6.1 Fault Management

Alarms are delivered through an SNMP flow to the Network Management System (NMS), allowing system administrators to detect faults and malfunctions in the node. Based on the alarm information, the system administrator can locate the source of the event and the relevant documentation, so that steps can be taken to resolve or prevent failures. Refer to *CUDB Node Fault Management Configuration Guide*, Reference [14] for more information about CUDB FM.

### 6.2 Performance Management

The elements of the CUDB node generate statistics and counters, enabling the operator to monitor the performance of each CUDB node. Statistics are collected by the internal performance components in each CUDB node and are made available to the NMS as standard 3GPP XML files. For more information on the related 3GPP standard, refer to *3GPP TS 32.435: Telecommunication Management; Performance Measurement: eXtensible*, Reference [24]. For further details on CUDB PM, refer to *CUDB Performance Guide*, Reference [15].

### 6.3 Log Management

The CUDB elements in the CUDB node generate log files, enabling the operator to monitor events on each CUDB node. Logs can be easily collected by an external management application using SFTP.

Authentication and command logs can be sent to an external logging server.



Refer to *CUDB Node Logging Events*, Reference [16] for more details on CUDB Logs.

## 6.4 Configuration and Software Management

Both NETCONF and a CLI interface are available for handling the configuration of each CUDB node. Configuration management is offered through a tree of Managed Objects with parameters that can be modified, or parameters that offer information on the status of certain components of CUDB (for instance, the status of a specific DS or whether a specific DS is master of its partition). CUDB includes tools to help define the initial Managed Objects tree and to verify if it is correct.

Certain CUDB operations are performed by executing ad-hoc commands.

Software inventory in a CUDB node shows the versions of each software package installed on each server of the CUDB node.

CUDB supports the execution of backups for the system configuration on a per CUDB node basis.

The LDAP Schema Management GUI ensures user-friendliness by providing a graphical interface to perform schema management operations (such as adding support for new applications). Refer to *CUDB LDAP Schema Management Graphical User Interface*, Reference [17] for further information.



## 7 Configuration

Configuration of the system has to take two levels into account: node level and system level. Both of these levels are described in the following sections.

### 7.1 CUDB Node Configuration

A CUDB node consists of a maximum of 36 blades per cabinet or 36 VMs per VNF.

There are two types of CUDB nodes:

- nodes with a PLDB
- nodes without a PLDB, only allowed if the “Flexible PL Deployment” function is used.

The components in the cabinet of nodes **with** PLDB are as follows:

- OAM servers: 2 blades or VMs.
- PLDB
  - Minimum size:
    - 4 blades for BSP 8100 (GEP3).
    - 2 blades for BSP 8100 (GEP5).
    - 2 VMs (virtual deployments).
  - Maximum size: 16 blades or VMs.
- DS units:
  - Minimum number of DS units: 1 DS unit, meaning 2 DS blades or VMs.
  - Maximum number of DS units: 15 DS units, meaning 30 DS blades or VMs.

The components in the cabinet of nodes **without** PLDB are as follows:

- OAM servers: 2 blades or VMs.
- DS units:
  - Minimum number of DS units: 2 DS units, meaning 4 DS blades or VMs.
  - Maximum number of DS units: 17 DS units, meaning 34 DS blades or VMs.

PLDB can be configured with any *even* number of servers between the minimum and maximum values. A larger PLDB size means that a lower number of DS units will fit in nodes with a PLDB.

**Note:** For CUDB systems deployed on native BSP 8100 (GEP5), when the "BSP Capturing Unit Option" is used, the subrack capacity is decreased by two blades. Refer to the BSP 8100 CPI for more information.

## 7.2 CUDB System Configuration

CUDB supports several different system configurations. Geographical redundancy can be set to either 1+1, or 1+1+1 modes. The maximum number of sites in a deployment is six. Figure 9 and Figure 10 shows examples for all deployment types.

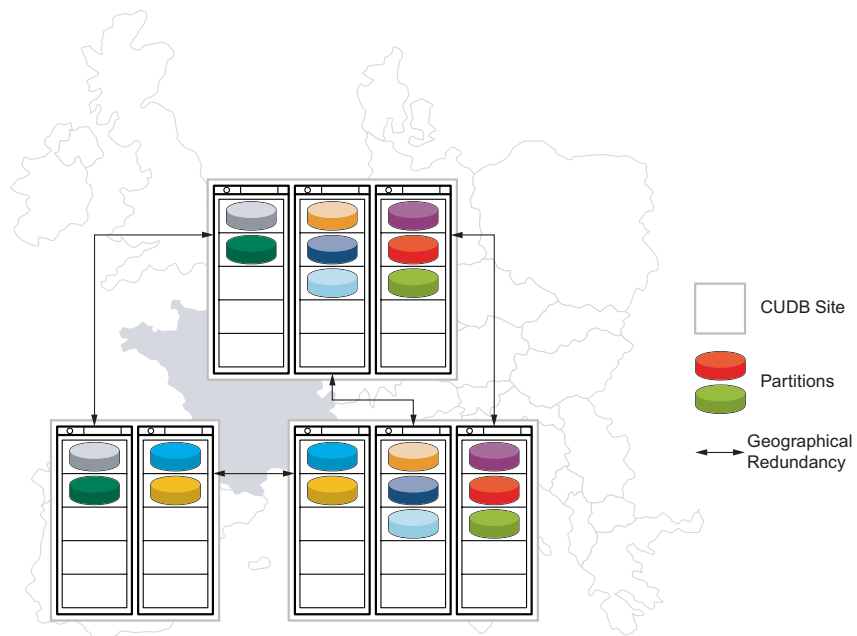
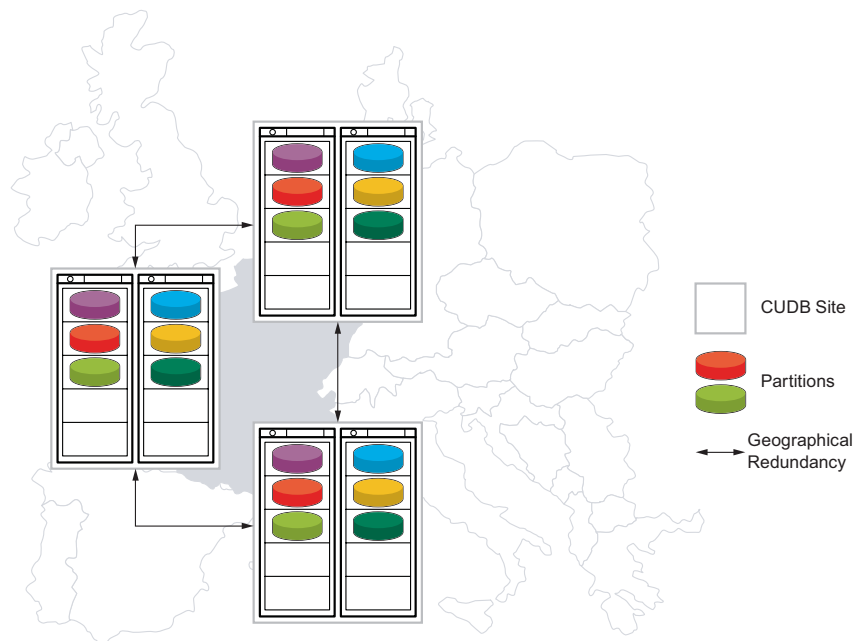


Figure 9 CUDB Deployment with Double Geographical (1+1) Redundancy



*Figure 10 CUDB Deployment with Triple Geographical (1+1+1) Redundancy*

Actual CUDB configurations must be set up according to operator requirements after network planning. Refer to *CUDB Deployment Guide*, Reference [18] for further details.





## 8 Characteristics

The following section highlight the most relevant characteristics of CUDB.

### 8.1 Availability and Reliability

The availability of CUDB is 99,999% at system level. The system is deployed with geographical redundancy, therefore the service is not interrupted, even if a whole site is down. This is because the other CUDB sites keep processing traffic, thereby providing non-stop, uninterrupted data access.

The CUDB architecture is also transparent from a data access point of view. Each CUDB node with a replica of PLDB provides access to the whole subscriber base, regardless of how data is distributed among the CUDB nodes. Data transparency is also supported by geographical redundancy: CUDB supports double or triple geographical redundancy (see Section 3.2.12 on page 12 for more information).

Every element is replicated at least twice inside a CUDB node. In this way, single points of failure are avoided, and maintenance operations (such as replacing a failing switch or network card) can be performed without stopping traffic. Therefore, CUDB is continuously available during SW and HW upgrades.

Data is replicated not only between the DS units of a partition, but also inside a DS cluster as well. Each DS cluster holds two copies of the data stored on the partition.

Periodical backups can be ordered for the whole system.

### 8.2 Scalability

CUDB has a high performance data layer that provides linear scalability both in capacity and performance, with bounded access latency. The CUDB system can increase the amount of stored data by adding additional DSGs. However, this operation does not require downtime, nor does it affect traffic in any way. DSGs can be added just by adding more server blades to the existing CUDB nodes, or by setting up additional CUDB nodes to the system.





# Glossary

**AAA**

Authentication, Authorization and Accounting

**ACL**

Access Control List

**AMC**

Automatic Mastership Change

**AUC**

Authentication Center

**BLOB**

Binary Large Object

**BSP**

Blade Server Platform

**CAS**

Customer Administration System

**CLI**

Command Line Interface

**CUDB**

Ericsson Centralized User Database

**DIT**

Directory Information Tree

**DS**

Data Store

**DSG**

DS Unit Group

**EBS**

Ericsson Blade System

**EGEM2**

Enhanced Generic Ericsson Magazine version 2

**EIR**

Equipment Identity Register

**EMA**

Ericsson Multi Activation

**ENUM**

E.164 Number Mapping

**EPC**

Evolved Packet Core

**FE**

Front End

**FM**

Fault Management

**GEP3**

Generic Ericsson Processor version 3

**GEP5**

Generic Ericsson Processor version 5

**GUI**

Graphical User Interface

**HLR**

Home Location Register

**HSS**

Home Subscriber Server

**HTTP**

Hypertext Transfer Protocol

**HTTPS**

Hypertext Transfer Protocol Secure

**IETF**

Internet Engineering Task Force

**IMS**

IP Multimedia Subsystem

**LDAP**

Lightweight Directory Access Protocol

**LDAPS**

LDAP over SSL



**MA**

Multi Activation

**MNP**

Mobile Number Portability

**NETCONF**

Network Configuration

**NMS**

Network Management System

**OAM**

Operation and Maintenance

**PG**

Provisioning Gateway

**PL**

Processing Layer

**PLDB**

Processing Layer Database

**PM**

Performance Management

**SASL**

Simple Authentication and Security Layer

**SCXB3**

System Control Switch Board version 3

**SFTP**

Secure File Transfer Protocol

**SLF**

Subscription Locator Function

**SNMP**

Simple Network Management Protocol

**SOAP**

Simple Object Access Protocol

**SSH**

Secure Shell

**SSL**

Secure Socket Layer

**TLS**

Transport Layer Security

**UDC**

User Data Consolidation

**UPG**

User Profile Gateway

**VNF**

Virtualized Network Function



## Reference List

### CUDB Documents

- [1] *CUDB LDAP Interwork Description*, 1/15519-HDA 104 03/9
- [2] *CUDB LDAP Data Access*, 5/15534-HDA 104 03/9
- [3] *CUDB Backup and Restore Procedures*, 1/1553-CNH 160 6130/9
- [4] *CUDB Import and Export Procedures*, 6/1553-HDA 104 03/9
- [5] *CUDB Notifications*, 9/15534-HDA 104 03/9
- [6] *CUDB Multiple Geographical Areas*, 13/15534-HDA 104 03/6
- [7] *CUDB Binary Large Object Attributes Management*, 2/15534-CRH 109 0494/6
- [8] *CUDB Subscription Reallocation*, 12/15534-HDA 104 03/9
- [9] *CUDB High Availability*, 7/15534-HDA 104 03/9
- [10] *CUDB Consistency Check*, 17/1553-CSH 109 067/9
- [11] *CUDB Security and Privacy Management*, 8/1553-HDA 104 03/9
- [12] *CUDB SOAP Interwork Description*, 3/15519-HDA 104 03/9
- [13] *CUDB Node Hardware Description*, 3/1551-CSH 109 067/9
- [14] *CUDB Node Fault Management Configuration Guide*, 3/1553-CSH 109 067/9
- [15] *CUDB Performance Guide*, 4/1553-HDA 104 03/9
- [16] *CUDB Node Logging Events*, 4/1553-CSH 109 067/9
- [17] *CUDB LDAP Schema Management Graphical User Interface*, 1/1553-CNH 160 6161/9
- [18] *CUDB Deployment Guide*, 2/1553-HDA 104 03/9
- [19] *CUDB LDAP Data Views*, 15/15534-HDA 104 03/9
- [20] *CUDB Glossary of Terms and Acronyms*, 0033-HDA 104 03/9

### Other Documents and Online References



- [21] *Lightweight Directory Access Protocol (LDAP)*. RFC 4510  
<http://www.rfc-editor.org/rfc/rfc4510.txt>
- [22] *The Secure Shell (SSH) Authentication Protocol*. RFC 4252  
<http://www.rfc-editor.org/rfc/rfc4252.txt>
- [23] *NETCONF Configuration Protocol*. RFC 4741 <http://www.rfc-editor.org/rfc/rfc4741.txt>
- [24] *Telecommunication Management, Performance Measurement: eXtensible Markup Language (XML) File Format Definition*. 3GPP TS 32.435  
<http://www.3gpp.org/ftp/Specs/html-info/32435.htm>