

CUDB Security and Privacy Management

USER GUIDE

Copyright

© Ericsson AB 2016-2018. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

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

Trademark List

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



Contents

1	Introduction	1
1.1	Purpose and Scope	1
1.2	Revision Information	1
1.3	Typographic Conventions	3
2	Overview	3
3	Network Security Configuration	4
3.1	Backbone Security	4
3.2	ICMP Reply	4
3.3	Allowed Protocols, Services, and Ports	5
3.4	Access Filtering	5
3.5	Switch Configuration	5
3.6	Router Configuration	5
4	User Management	6
4.1	LDE User Management	6
4.1.1	cudbadmin Password Expiration Policy	7
4.1.2	User Policies	7
4.1.3	Password Strength Policy	10
4.1.4	General Password Recommendations	11
4.2	Remote User Authentication Management	12
4.2.1	Constraints of Remote User Authentication	15
4.2.2	List of Local CUDB Users and Groups, Not Valid on Remote Servers	16
4.2.3	Examples of Using Remote User Authentication	17
4.3	LDAP User Management	20
4.3.1	Changing the Password of the rootdn User	20
4.3.2	Access Control Configuration Rules	21
4.3.3	Storage of LDAP users passwords	25
4.4	Switch User Management	27
4.4.1	Remote Switch User Authentication Management	27
4.5	Router User Management	27
5	System Audit Logging	28
5.1	LDE Audit Logging	28
5.2	Switch Audit Logging	28
5.3	Router Audit Logging	29



6	Certification Authority	29
7	Certificate Generation	30
8	Configuring Secure Database Replication	31
9	Configuring Secure Notifications	33
9.1	TLS Configuration	33
10	Configuring LDAP Front End Security	35
10.1	LDAP Authentication Types	35
10.2	Configuring TLS for LDAP Front End	35
10.3	Configuring Secure LDAP Proxy	37
11	Configuring Secure CUDB OAM Centralized Authentication System Support	37
12	Configuring Secure Centralized Security Event Logging	39
12.1	Examples of Using Secure Centralized Security Event Logging	40
12.1.1	Using TLS	40
12.1.2	Not Using TLS	41
13	Other Security Recommendations	42
14	Privacy	43
14.1	Notice	43
14.2	Consent	44
	Glossary	45
	Reference List	47



1 Introduction

This document provides a description for the security functions available in the nodes of the Ericsson Centralized User Database (CUDB).

1.1 Purpose and Scope

The purpose of this document is to provide an overview of the CUDB security functions available in the nodes, and describe the guidelines of configuring the system to the security level requested by CUDB operators.

1.2 Revision Information

Rev. A

This document is based on 8/1553-HDA 104 03/9 with the following changes:

- Removed obsolete information and updated references accordingly.
- Added information on a cloud infrastructure and marked BSP 8100 information throughout the document.
- Section 3.3 on page 4: Renamed and updated section.
- Section 4.1 on page 6: Updated user group description.
- “OAM Automation with NETCONF Support” updates throughout the document.
- Section 4.1.2.3 on page 7: Added new subsection.
- Section 4.1.2.4 on page 8 and Section 4.1.2.5 on page 9: Added `grep` command and updated uses of `tail` command.
- Section 4.1.2.5 on page 9: Updated basic ESA commands section.
- Section 4.1.3.3 on page 11: Updated rules of strong passwords.
- Section 4.3.1 on page 20: Added a Warning about ASCII alphabetic and numeric digit characters.
- Section 4.3.3.2 on page 26: Updated Step 2 with the `updateUserInfo` administrative operation.
- Section 6 on page 29: Updated with note on backward compatibility with SSL.



- Section 13 on page 42: Updated with new recommendation on performing scheduled maintenance actions.

Rev. B

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

- Section 4.3.2 on page 21: Updated ACLs Configuration steps.
- Section 7 on page 30 and Section 10.3 on page 36: Updated VIP names.

Rev. C

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

- Section 4.1 on page 6, Section 4.1.2.5 on page 9, Section 4.2 on page 12: Updated the name of `cudbOperators` group.
- Section 4.2 on page 12: Updated with new `tlsMode` attribute.

Rev. D

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

- **Section 4.1.2.1 Configuring Idle Session Timeout:** Removed obsolete section.
- Section 4.2 on page 12, Section 4.2.1 on page 15, and Section 4.2.3 on page 17: Added information regarding the login shell and updated `loginShell` rows in Example 1.

Rev. E

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

- **Section 4.6 UDC Cockpit User Management:** Removed obsolete section.
- Section 12 on page 38: Updated description stating that TCP is supported.

Rev. F

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

- Section 4.3.3.2 on page 26: Updated note.

Rev. G

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

- Section 12.1 on page 40: Added Section 12.1.2 on page 41.



Rev. H

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

- Section 4.2 on page 12: Updated `cudbOperators` and `cudbadmin` groups.
- Section 4.2.1 on page 15: Updated descriptions.
- Section 4.2.2 on page 16: Updated description, removed note.
- Section 4.2.3 on page 17: Added note, updated `cudbOperators` local group, Example 1, and Example 3.

Rev. J

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

- Section 4.3.1 on page 20, Section 6 on page 29, Section 9 on page 32, Section 10.2 on page 35, Section 12 on page 38: Updated admonitions.
- Section 10.1 on page 35: Updated title and description.

1.3 Typographic Conventions

Typographic conventions can be found in the following document:

- *Typographic Conventions*

2 Overview

This document provides instructions on how to secure CUDB network interfaces and the CUDB node itself to protect the system from malicious attacks and prevent unauthorized access to the information stored in the system.

The document offers security guidelines for the following topics:

- Network configuration.
- Node administration and user management.
- Transport Layer Security (TLS) traffic encryption and authentication between CUDB nodes.



- Secure notifications.

The topics listed above are described in the following sections in more detail.

This document also provides the CUDB privacy-related statements.

3 Network Security Configuration

This section describes the security configuration related to communication between different nodes. For further information about network configuration, refer to *CUDB Node Network Description*, Reference [1].

3.1 Backbone Security

The CUDB backbone is protected by the following means:

- Secure database replication (see Section 8 on page 31).
- Secure Lightweight Directory Access Protocol (LDAP) proxy configuration (see Section 10.3 on page 36 for more information).

Note: CUDB also supports IPSec, used to encrypt communication links. IPSec configuration is however not in the scope of this document.

CUDB offers a large set of security functions to secure communication channels. The use of these functions depends on the security level that CUDB operators prefer to reach.

3.2 ICMP Reply

The Internet Control Message Protocol (ICMP) is allowed by default, since its use does not pose any risks, as CUDB is not exposed to external networks. However, to avoid CUDB being flooded with ICMP messages, other network infrastructure elements (such as routers) must be configured accordingly.

For CUDB systems deployed on native BSP 8100, consult the next level of maintenance support for more information on how to define a policy to disable ICMP traffic in the Component Main Switch (CMX).

For CUDB systems deployed on a cloud infrastructure, refer to the routing related documentation of the infrastructure to determine how to disable ICMP.



3.3 Allowed Protocols, Services, and Ports

Refer to *CUDB Node Network Description*, Reference [1] for detailed information about the protocols, services and ports that must be open or allowed in order to perform CUDB operations.

3.4 Access Filtering

Network security level may be increased by implementing proper access filtering.

For CUDB systems deployed on native BSP 8100, CMX routers implement advanced Policy Based Forwarding (aPBF) rules. These aPBF functions allow to configure or filter match rules and action policies for CMX. Consult the next level of maintenance support for further details.

For CUDB systems deployed on a cloud infrastructure, refer to the routing related documentation of the infrastructure for information about the available access filtering tools and policies.

3.5 Switch Configuration

Implementing secure access to the switches is recommended.

For CUDB systems deployed on native BSP 8100, the System Control Switch Board (SCXB) can be configured by means of a Secure Shell (SSH) interface. If needed consult the next level of maintenance support.

For CUDB systems deployed on a cloud infrastructure, refer to the switching related documentation of the infrastructure for information about implementing secure access to the switches.

3.6 Router Configuration

This section provides information on how to increase security with proper router configuration.

For CUDB systems deployed on native BSP 8100, the CMX control interfaces can be either trusted or non-trusted, which means the following:

- Control interfaces towards the backplane are considered trusted and are open at system start by default. On trusted ports, CMX supports both secure and non-secure protocols, like telnet, TFTP, SNMPv2c, SNMPv3, or DHCP.
- Interfaces on the front are considered non-trusted and are closed at system start by default. On non-trusted ports, CMX supports secure protocols only (like SSH, SCP) and DHCP.



The control interfaces work according to the following security restrictions:

- The front management interface (combined debug connector for RS232 and eth2) is disabled by default at CMX startup.
- `Root` access is blocked for login. Users must login as `Advanced` users first and then make a `su` login as `root`. See Section 4.5 on page 27 for more information on the default CMX users.
- Telnet is allowed only for the backplane interfaces.

For CUDB systems deployed on a cloud infrastructure, refer to the routing related documentation of the infrastructure for information about securing routers configuration.

4 User Management

CUDB offers the following user management tools from a security perspective:

- Linux Distribution Extensions (LDE) user management.
- LDAP user management.
- Router and switch user management.
- Remote user authentication management.

These tools are described in the following sections in more detail.

4.1 LDE User Management

The CUDB users are based on LDE, and are managed on node level. This means that all servers of the CUDB node see the same set of existing users.

The users are stored persistently, which means that they are not deleted if the server is rebooted.

CUDB is preconfigured with two user groups:

- `cudbadmin`: This group contains a single user with the same name (`cudbadmin`), who can perform all configuration tasks with the `sudo` command.
- `cudbOperators`: This group also contains a single user with the same name (`cudbOperator`), who can perform certain configuration and maintenance tasks, as further detailed below.



Additional users and user groups can also be created. It is mandatory to define passwords for all the groups and to include all the defined users into a group.

Refer to *CUDB Users and Passwords*, Reference [2] for more information on the default user and password pairs used in the CUDB system. For more information on LDE, refer to *LDE Management Guide*, Reference [8].

4.1.1 **cudbadmin Password Expiration Policy**

The password expiration policy of LDE users is configured with the `chage` command.

The default password validity of `cudbadmin` is 30 days, with password change reminders set to be raised 10 days before password expiration.

4.1.2 **User Policies**

CUDB offers several user account settings to enhance the security of session establishment. The steps of configuring these settings is described below.

4.1.2.1 **Configuring the Maximum Number of Failed Login Attempts**

To configure the maximum number of login attempts, set the value of the `maxNumFailedLogins` attribute located under the `cudbSystemSecurity` class. For more information, refer to the *Class CudbSystemSecurity* section in *CUDB Node Configuration Data Model Description*, Reference [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).

4.1.2.2 **Configuring Lockout Period**

To configure the user lockout period, set the value of the `lockoutPeriod` attribute located under the `cudbSystemSecurity` class. For more information, refer to the *Class CudbSystemSecurity* section in *CUDB Node Configuration Data Model Description*, Reference [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).

4.1.2.3 **Configuring Inactivity Timer**

By default, an inactivity timer is enabled for each user account. Account expiry is moved forward by (a fixed) 90 days on each login.



4.1.2.3.1 Accounts not Expiring

Accounts which should not expire must belong to the group `no_expire`. This privilege can be set by the CUDB system administrator:

1. Establish a new administrative CUDB CLI session towards one of the System Controller (SC) blades of the CUDB node:

```
ssh <admin_user>@<CUDB_Node_OAM_VIP_Address>
```

2. After creating a user, execute the following commands:

```
SC_2_1# sudo /usr/sbin/usermod <user_name> -aG no_expire  
-e -1
```

4.1.2.3.2 Expired Accounts

If a user account is expired, it can be re-enabled by the CUDB system administrator:

1. Establish a new administrative CUDB CLI session towards one of the SC blades of the CUDB node:

```
ssh <admin_user>@<CUDB_Node_OAM_VIP_Address>
```

2. Execute the following command to re-enable the account:

```
SC_2_1# sudo /usr/sbin/usermod <username> -e  
"<YYYY>--<MM>--<DD>"
```

In the command above, `<YYYY><MM><DD>` stand for the year, the month and the day of the renewed expiry date of the user account.

By default, the root user account and the following pre-defined local CUDB user accounts will never expire:

- `cudbOperator`
- `cudbadmin`
- `mysql`
- `cluster_supervisor`
- `system_monitor`
- `open_ldap`
- `g_soap`

Note: When a user account is newly created, no account expiry date is set until the first login of the user.



4.1.2.4 cudbadmin Group Permissions

The `cudbadmin` group permissions are assigned by using the `/etc/sudoers` file. Permissions can be granted as follows:

- Creating a list of commands that `cudbadmin` group users can execute through `sudo`.
- Creating a list of the directories where the assigned commands reside.
- Both of the above.

By default, the CUDB system is preconfigured with the following paths in the `/etc/sudoers` file:

- `/opt/ericsson/cudb/mysql/libexec/`
- `/opt/ericsson/cudb/OAM/bin/`
- `/opt/ericsson/cudb/CUDBOI/bin/`
- `/opt/ericsson/cudb/ESA/bin/`
- `/opt/ericsson/cudb/Monitors/bin/`
- `/opt/coremw/bin/`
- `/opt/com/bin/`
- `/usr/bin/comsa*`
- `/opt/ericsson/cudb/mysql/bin/`
- `/opt/ericsson/cudb/ldapfe/bin/`

Users in this group can read log files on the `/var/log/SC_2_*` and `/var/log/PL_2_*` paths by using the `tail` and `grep` commands, as shown below:

```
sudo grep ERROR /var/log/PL_2_3/messages
```

```
sudo tail -50 /var/log/SC_2_1/kernel
```

Note: For security reasons, `cudbadmin` group users have no write access in the above directories.

4.1.2.5 cudbOperators Group Permissions

The `cudbOperators` user group is used to perform common configuration and maintenance tasks. Users of this group can perform the following tasks as super users:

- Run basic CUDB OAM commands, such as `cudbSystemStatus`, as shown below:



```
sudo /opt/ericsson/cudb/OAM/bin/cudbSystemStatus
```

- Run basic ESA commands, detailed in *ESA Performance Management, Reference* [9].

- Read system log messages, as shown below:

```
sudo tail -f /var/log/PL_2_3/messages
```

```
sudo grep ERROR /var/log/PL_2_3/messages
```

- Read the normal and error log messages of the LDAP Front Ends (FEs), as shown below:

```
sudo tail -f /var/log/ldapfe
```

```
sudo tail -f /local2/log/ldapfe_errors
```

- Read the alarms log, as shown below:

```
sudo cat /var/log/ESA/alarms.log
```

- List the contents of the `/local/cudb/mysql/ndbd/backup/` directory and each directory residing under it. This directory hierarchy contains generated data backup files of the database cluster. For example:

```
sudo ls -l /local/cudb/mysql/ndbd/backup/
```

Refer to *CUDB Node Logging Events, Reference* [4] for more information on CUDB logs.

4.1.3 Password Strength Policy

When new user accounts are created, the CUDB system applies the default password settings on them. It is recommended to configure the password strength according to the desired behavior by following the below instructions.

4.1.3.1 Configuring the Minimum Password Length

To configure the minimum password length, set the value of the `minPasswordLength` attribute located under the `cudbSystemSecurity` class. For more information, refer to the *Class CudbSystemSecurity* section in *CUDB Node Configuration Data Model Description, Reference* [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description, Reference* [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).



4.1.3.2 Configuring the Minimum Number of Unique Passwords

To configure the minimum number of unique passwords before an already used password can be selected again, set the value of the `minPasswordNonRepeat` attribute located under the `cudbSystemSecurity` class. For more information, refer to the *Class CudbSystemSecurity* section in *CUDB Node Configuration Data Model Description*, Reference [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).

4.1.3.3 Configuring Password Strength

To ensure the use of strong passwords, the following rules apply to password configuration:

- The password must be at least eight characters long.
- The password must contain at least three special characters from the following five character groups:
 - Lowercase letters
 - Uppercase letters
 - Numbers (such as 1, 2, 3, and so on)
 - Symbols (such as @, =, -, and so on)
 - Unicode characters
- The password must not contain more than three equal, consecutive characters.
- Any new password must differ from the previous one in at least three characters.
- The password must not be the same as the user ID or its reverse wording.

Note: These restrictions do not apply to the root user. When configuring a new root password which does not meet the above restrictions, the system sends a warning message, but allows changing the password.

4.1.4 General Password Recommendations

CUDB users are strongly advised to abide by the following password handling recommendations:

- Keep passwords confidential.



- Avoid keeping record of the passwords either on paper, or in digital format (such as in software files, hand-held devices, and so on), unless the list can be stored securely, and it has been approved to do so.
- Change passwords whenever there is any indication that the system or a password is compromised.
- Do not include passwords in any automated logon process (for example stored in a macro or function key), unless it is strictly necessary.
- Use the lowest level access account to perform an action. For example, do not use the `root` account unless the action requires `root` access.
- If shared accounts are used, users must login with their own account, and then switch to the user account (such as the `root` user) that must be used to perform an action. User switch is initiated with the `su` or `sudo` command.

4.2 Remote User Authentication Management

CUDB OAM Centralized Authentication System Support provides a mechanism to authenticate external OAM users stored in a remote LDAP authentication server, not defined locally, and grants them access to OAM SSH interfaces in CUDB nodes. This allows to define and manage new users and groups only in remote LDAP servers, in a centralized way, instead of having to create them locally on all CUDB nodes. Locally defined OAM users are still authenticated locally.

Remote users created inside an LDAP authentication server can belong either to remote groups created in an LDAP authentication server, or to local groups defined inside CUDB nodes, for example the `cudbOperators` or `cudbadmin` groups. Authorization rules (permissions) for those groups are applied depending on the context of definition as follows:

- In case the added remote users belong to a local group, `sudoers` rules that apply to the local group also apply to the remote user (see Section 4.1.2.4 on page 8 and Section 4.1.2.5 on page 9 for more details).
- In case a remote user is created belonging to an added remote group, by default its permissions correspond to others on each file and folder of the file system. This means that, by default, the permissions are limited: for example, the user is unable to execute CUDB commands, see LDAP logs, and so on. However, it is possible to modify `sudoers` locally inside the CUDB nodes to set new rules that apply to any added remote group, thereby making these groups gaining some, all, or even more permissions than the `cudbadmin` or `cudbOperators` user groups. See Section 4.2.3 on page 17 for an example of this configuration.

Note: `sudoers` modifications require root privileges, and therefore must be performed by Ericsson support.



It is possible to configure one or two remote LDAP authentication servers, so in case the first one is temporarily down, the second server can provide the authentication service.

The following mandatory parameters must be configured:

- The remote IP address of the primary LDAP server (at least one server is mandatory, the second one is optional).
- The `baseDn`, to specify where to look for new users and groups inside the remote LDAP authentication servers.

Optionally, the following additional configuration attributes may need to be set:

- `bindDn` and `bindPassword`, to limit the access to the LDAP server.
- `tlsEnabled`
- `tlsMode`
- The secondary server IP.

When configuring Centralized Authentication, CUDB, as an additional and unconditional step, overrides the login shell for remote users by putting the `nss_override_attribute_value loginShell /bin/bash` directive line in `/home/cudb/security/config/ldapAA_ldap.conf`.

Refer to *CUDB Node Configuration Data Model Description*, Reference [3] for more information on configuring these parameters on the `CudbExternalAuthServer` class.

CUDB OAM Centralized Authentication System Support supports connection to remote LDAP authentication servers that either have TLS disabled (so messages between CUDB node and remote LDAP authentication servers are not cyphered), or enabled (so messages are cyphered). By default, TLS is disabled: to enable it, set the proper certificates before activation by following the steps of Section 11 on page 37.

If TLS is enabled, the Certification Authority certificate must be part of the file set in the `tlsCaCertificatesFile` attribute of the `CudbSystemSecurity` class.

It is also possible to configure the method of establishing a secure connection to the external server using the `tlsMode` attribute.

- **STARTTLS:** *Opportunistic TLS* mode is used. Secure sessions are set up through the same port that is used for non-secure sessions, port 389. This is the default setting.
- **LDAPS:** Secure sessions are set up through the dedicated port, port 636.



Once the certificates are configured, set the `tlsEnabled` attribute to `true`. Refer to *CUDB Node Configuration Data Model Description, Reference* [3] for more information on configuration.

Once the `CudbExternalAuthServer` class is configured, enable external authentication by setting the `enabled` attribute of the `CudbExternalAuthMgmt` class to `true`.

When configuring and enabling the function, consider also the following:

- The `enabled` attribute of `CudbExternalAuthMgmt` cannot be set to `true` if the `CudbExternalAuthServer` instance does not exist.
- The instance of the `CudbExternalAuthServer` class cannot be deleted until the feature is disabled (that is, the `enabled` attribute of the `CudbExternalAuthMgmt` class is set to `false`).
- The configuration of the `CudbExternalAuthServer` class can be modified by setting the `enabled` attribute of `CudbExternalAuthMgmt` to `true` or to `false`.
- Introduce the changes in `CudbExternalAuthMgmt` and `CudbExternalAuthServer` by using different commits.
- Execute the `applyConfig` administrative operation after each commit. Do not include two or more commits in the same `applyConfig` execution.
- The changes are not applied until the `applyConfig` action is executed.

Once properly configured and enabled, SSH connections to a CUDB node can be done by a remote or a local user. A remote user is a user that is created on the remote LDAP authentication servers; a local user is defined in the CUDB node. If an SSH connection to a CUDB node is initiated, different scenarios can occur based on the availability of the authentication servers. These scenarios are as follows:

- The CUDB node first tries to authenticate the user on the first remote LDAP authentication server. For this process, the destination CUDB acts as an LDAP client, and connects to the remote LDAP authentication server to verify the provided password.
- If the first server is unavailable, the same operation is repeated with second remote LDAP authentication server.
- If no remote servers are available, the CUDB node looks for the user locally.
- If at least one of the servers is available, but the user is not defined on the remote LDAP authentication server, the user is not a remote user. Accordingly, the CUDB node looks for the user locally.
- If at least one of the servers is available, and the user is defined on the server, and is, as such, a remote user, but the given password is incorrect, the operation fails, even if the same user is also defined locally with a



different password that matches the locally entered password, but not the remote one.

- If authentication using a remote user is not possible, local authentication must be performed instead, using a different local user, not defined on any remote LDAP servers.

4.2.1

Constraints of Remote User Authentication

The following limitations are applied to **CUDB OAM Centralized Authentication System Support**:

- This function is not compatible with the COM LDAP Authentication configuration. Therefore, the following configuration model element must be kept set to `LOCKED`:

```
ManagedElement=1, SystemFunctions=1, SecM=1, UserManagement=1, LdapAuthenticationMethod=1, administrativeState=LOCKED
```

If set to `UNLOCKED`, the LDE authentication configuration (PAM and NSS) set by this function can overwrite, or be overwritten by the configuration set in COM LDAP Authentication.

- A new user created on remote LDAP authentication servers must have a different username and a different `uidNumber` than any local user. Do not create remote users which override existing local CUDB ones, and also do not add a new user at the same time locally on CUDB nodes and on remote LDAP authentication servers. By ignoring this limitation, and having a user that exists in both locations with the same username, authentication will not work with the local password, but only with the password stored on the remote LDAP server. Only if the LDAP servers are unavailable will the local password work.

Note: If a user is duplicated (that is, the username is the same on both locations), but the locally and remotely defined attributes (that is, the `uidNumber`, `gidNumber`, and type of shell to use) are different, then the local definition of those attributes are taken into account. This never applies to the password: the remote one is the always correct for login.

- For security reasons, the previous limitation is more restrictive to duplicated default CUDB local users. Without exception, any definition of these default local CUDB users or default local groups on remote LDAP authentication servers is completely ignored. Only local passwords and local definitions of those users and the group attributes are applied. See Section 4.2.2 on page 16 for the list of these users and groups.
- Remote users created in the LDAP authentication servers must have a user ID (`uidNumber` attribute), which is not already used for local users, to avoid collision with them. Otherwise, remote user will fail on authentication,



even if the password is correct. Local users defined in the system can be checked by issuing the `awk -F':' '{ print $1}' /etc/passwd` command.

- Remote groups created in LDAP authentication servers must have a group ID (`gidNumber` attribute), which is not already used for local groups, to avoid collision with them. Local groups defined in the system can be checked in the `/etc/group` directory.
- Remote users can belong to any remote or local groups, except the root group. Because of this limitation, any remote user with `gidNumber 0` automatically fails on authentication, even if the password is correct.
- CUDB overrides the login shell for remote users and sets it to `/bin/bash`.
- If the LDAP servers configured on the CUDB nodes are down, the CUDB node performance will not be impacted, only remote user authentication will be unavailable. However, if the configured IP addresses of the LDAP servers are unreachable IPs, a delay of a few seconds can happen on the SSH connections of local users, as each SSH will try to connect to those IP addresses first.

4.2.2 List of Local CUDB Users and Groups, Not Valid on Remote Servers

To avoid conflict, do not use the following predefined local CUDB usernames to create a new remote user on the LDAP authentication servers:

- `root`
- `cudbOperator`
- `cudbadmin`
- `mysql`
- `cluster_supervisor`
- `system_monitor`
- `open_ldap`
- `g_soap`
- `ntp`
- `dhcpd`

If any of the usernames above is reused as the username of a new remote user created on the LDAP authentication servers, the username on the remote server will be ignored: only the local definition of the user and the local password will work.



Regarding groups, any remote group created on the LDAP authentication servers with a `gid` corresponding to a default CUDB local group will be ignored, only the local group definition will apply. However, a remote user can belong to local groups (except to 0, as described in Section 4.2.1 on page 15). Local groups defined in the system can be checked in `/etc/group` directory.

4.2.3 Examples of Using Remote User Authentication

In the following example, OpenLDAP is used for the remote LDAP authentication servers, but any LDAP server compliant with [RFC 2307: An Approach for Using LDAP as a Network Information Service](#), Reference [12] is suitable, ensuring that the required object classes and attributes for storing users and groups information are used (mainly `posixAccount` and `posixGroup` classes). In case of OpenLDAP, the following schemas are used:

- `nis.schema`
- `core.schema`
- `cosine.schema`

After the remote LDAP authentication servers are properly configured and started, new remote groups and remote users can be defined. Example 1 shows how to define one new remote group and two new remote users in LDAP authentication servers. One user belongs to the newly-defined remote group (9999) and the other belongs to the already existing local group `cudbOperators`.

Note: Check local group `gid` in `/etc/group` before creating a new remote user which belongs to `cudbOperators` group. In Example 1 and Example 3 below, `cudbOperators` `gid` is 1007.



```
# remoteAdminGroup, Groups, ericsson.com
dn: cn=remoteAdminGroup,ou=Groups,dc=ericsson,dc=com
objectClass: top
objectClass: posixGroup
cn: remoteAdminGroup
gidNumber: 9999
description: remote admin group

# user1Admin, Users, ericsson.com
dn: uid=user1Admin,ou=Users,dc=ericsson,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: posixAccount
uid: user1Admin
cn: user1
sn: admin
title: aaaaa1
telephoneNumber: +34 000 000 0001
loginShell: /dev/null
homeDirectory: /home/user1Admin/
description: user1 administrator user
userPassword:: e1NTSEF9azBPY2o1ankrUHEzcXR4dDlrU0JwQjI2K1J5NjVJb1I=
uidNumber: 1500
gidNumber: 9999

# user2Oper, Users, ericsson.com
dn: uid=user2Oper,ou=Users,dc=ericsson,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: posixAccount
uid: user2Oper
cn: user2
sn: oper
title: aaaaa2
telephoneNumber: +34 000 000 0001
loginShell: /dev/null
homeDirectory: /home/user2Oper/
description: user2 operator user
userPassword:: e1NTSEF9dUFkQVhmMVgyaXdLVE1xM3RObHZUaG1ZUm1UREVXdmcE=
uidNumber: 1700
gidNumber: 1007
```

Example 1 Defining a New Remote Group and Two New Remote Users

On a CUDB node, the **CUDB OAM Centralized Authentication System Support** function must be properly configured in the configuration model element as follows:

- At least one IP must be set for a remote LDAP authentication server.
- `baseDn` must be set. In Example 1 it is set to `dc=ericsson,dc=com` as both users and groups are contained in that element.
- `enabled` is set to `true`.

After the above configuration is properly done, execute the following steps to log in:

1. Execute the following command:

```
ssh user1Admin@<CUDB_node_ip>
```

2. Enter the proper password stored in the `userPassword` attribute.



Note: Hashed storage is recommended, for example with `slappasswd` in case of using OpenLDAP.

If both steps are successful, the `user1Admin` user successfully logs into the CUDB node. The specified `/home/user1Admin/` home directory is then created as shown in Example 2, in case it did not exist before.

```
CUDB_xx SC_2_1# ssh user1Admin@10.22.0.1
Password:
Creating directory '/home/user1Admin/'.
Last login: xxxxxxxxxxxxxxxx
SC_2_1 user1Admin/> id
uid=1500(user1Admin) gid=9999(remoteAdminGroup) groups=9999(remoteAdminGroup)
SC_2_1 user1Admin/> echo $SHELL
/bin/tcsh
```

Example 2 Successful Login with user1Admin

The same applies to the `user2Oper` user as shown in Example 3. As that user belongs to the `cudbOperators` group, it can execute the same `sudo` commands that any `cudbOperator` user can.

```
CUDB_xx SC_2_1# ssh user2Oper@xx.xx.xx.xx
Password:
Creating directory '/home/user2Oper/'.
Last login: xxxxxxxxxxxxxxxx
CUDB_xx SC_2_1$ id
uid=1700(user2Oper) gid=1007(cudbOperators) groups=1007(cudbOperators)
CUDB_xx SC_2_1$ echo $SHELL
/bin/bash

CUDB_xx SC_2_1$ /usr/bin/tail -f /var/log/SC_2_1/messages
/usr/bin/tail: cannot open `/var/log/SC_2_1/messages' for reading: Permission denied

CUDB_xx SC_2_1$ sudo /usr/bin/tail -f /var/log/SC_2_1/messages
<works>
```

Example 3 Successful Login with user2Oper

For users not belonging to local groups but to new remote groups, it is possible to modify the local `sudoers` file to allow it to have the rules of some local groups.

For example, `sudoers` can be modified to make the new remote group `remoteAdminGroup` have the same rules as the local `cudbadmin` group, by running as root `visudo` and changing the line `User_Alias ADMINS = %cudbadmin` to `User_Alias ADMINS = %cudbadmin,%remoteAdminGroup`.

After saving, the change is persistent to reboots, as it is stored under `/cluster` and is synchronized with all CUDB node blades or VMs.

Other adaptations can also be done in `sudoers` to set new rules for new remote groups or specifically for some users.



4.3 LDAP User Management

The CUDB LDAP users represent application FEs, and can read or potentially modify certain branches of the provisioned data in CUDB. User privileges can be set by the CUDB system administrator by changing the Access Control List (ACL) rules. LDAP users must be authenticated with the LDAP BIND operation using a configured password. Simple and Simple Authentication and Security Layer (SASL) authentication is supported.

Note: `rootdn` has write access to any part of the Directory Information Tree (DIT), and its access cannot be restricted with ACLs.

4.3.1 Changing the Password of the `rootdn` User

Attention!

This procedure must only be performed by authorized Ericsson personnel.

The LDAP password of the traffic root Distinguished Name (DN) can be updated for security reasons in the CUDB Configuration Data Model. To update the LDAP password, modify the value of the `ldapRootPassword` attribute located under the `CudbLdapAccess` class. For more information, refer to the *Class CudbLdapAccess* section in *CUDB Node Configuration Data Model Description*, Reference [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).

Do!

The password of the `rootdn` user can contain only ASCII alphabetic, numeric digit characters, and the following symbols: `,-%=?+~_`

It is recommended to follow the general password recommendations described in this document. For further information, see Section 4.1.4 on page 11.

Note: Connections opened towards LDAP FEs are lost during password change, since the LDAP FEs are restarted to reload the new password.



Attention!

Changing the password of the **rootdn** user can impact the traffic in the CUDB system, due to the restart of the LDAP FEs, and because CUDB nodes where password change has not yet taken place reject traffic.

4.3.2 Access Control Configuration Rules

This section describes the rules of configuring access control settings. As stated before, ACLs may be configured to set the privileges for every LDAP user to access the LDAP DIT.

ACL processing impacts performance: the larger the list of rules, the more time it takes to process. Therefore, it is recommended to create short rule sets by grouping users.

Note: Apply ACL rules with care, since changes can result in unexpected access denials due to the access control rule syntax and semantics. Always test changes on a test configuration before live deployment.

Follow the steps below to configure access control rules for a CUDB node:

1. Get the existing `acls.conf` file from `/cluster/home/cudb/dataAccess/ldapAccess/ldapFe/config/`

```
scp <admin_user>@<CUDB_Node_OAM_IP_Address>:/cluster/home/cudb/dataAccess/ldapAccess/ldapFe/config/acls.conf
acls.conf
```

2. Open the file for editing.

There are two CUDB system mandatory rules in the file: the first one and the last one; and there might be other application specific rules. All these rules must be kept.

3. Insert specific application rules between them and carefully consider adding other existing application specific ones, so they are ordered from less generic to more generic.

```
access to dn.children="ou=admin, <cudbRootEntryDn>"
by * auth

# APP SPECIFIC RULE SECTION
#

access to *
by users write
```



Note: `<cudbRootEntryDn>` in rule file must be replaced with corresponding CUDB `baseDn` value.

4. Copy the edited `acls.conf` file to the CUDB node with the following command:

```
scp acls.conf <admin_user>@<CUDB_Node_OAM_IP_Address>:/home/cudb/oam/configMgmt/commands/tmp/acls.conf
```

5. Establish a “CUDB CLI” session with the following command towards the CUDB node where the ACLs are going to be changed:

```
ssh <admin_user>@<CUDB_Node_OAM_IP_Address>
```

6. Apply the configuration with the `applyConfig` administrative operation. For more information, refer to the *Modification Procedure Using CUDB Configuration CLI* section in *CUDB Node Configuration Data Model Description*, Reference [3].

Depending on the syntax of the configuration file, the operation can have the following outcomes:

- If the modified `acls.conf` file has correct syntax, it is deleted from the directory where it was copied, and the new ACLs are applied to the CUDB node.
- If the modified `acls.conf` file does not have the right syntax, it is renamed to `acls.conf.bad`, and no changes are made to the ACLs currently used on the node.

7. Exit from the “CUDB CLI” session with the following command:

```
exit
```

Note: The ACLs must be the same in the entire CUDB system. Therefore, if the access control rules are changed in a node, the procedure described above must be executed on every node of the CUDB system starting from Step 4 (copying the modified `acls.conf` file to every node, then following the rest of the steps).

4.3.2.1

Access Control Configuration Guidelines

This section provides some guidelines on creating ACLs for CUDB. The guidelines are based on the contents of the OpenLDAP manual: refer to *OpenLDAP Software 2.4 Administrator's Guide*, Reference [10], for specific information on the ACLs.

Note: The longer the access control list, the more it degrades the performance of the node. It is advised to keep the number of users (and therefore the number of rules) to a minimum.

Access control rules follow the below syntax:



access to <what> by <who> [by <who> ...] [break]

A rule can span a number of lines, provided that the first character of all additional lines is a blank space. The elements of the above syntax are as follows:

- **<what>**

<what> refers to a branch, or optionally to attributes, object classes, or both. Both the attributes and object classes are specified with the **attrs=** clause as shown in the examples below:

```
dn.subtree="ou=multiSCs,dc=operator,dc=com"
dn.regex="serv=auth,mscId=" attrs=AuthIMSIData,AuthSubscriberData
dn.regex="serv=ims,mscId=" attrs=CSLOC,VLRADD,MSCARF
```

Note: **attrs** can indicate the list of attributes, object classes, or both. This is because an object class name is essentially a shorthand for all the attributes of the object class.

- **<who>**

<who> is a user name, standing either for the root DN, or a provisioned user in the **ou=admin** branch as shown in the below example:

```
dn.exact="cudbUser=auc,ou=admin,dc=operator,dc=com"
```

- **break**

The **break** keyword is used to force reading the access control list even if the requested DN matches the name defined with the **<what>** variable. Consider the following when using this command:

- **break** decreases system performance when added to a rule. Therefore, avoid its use if possible. An alternative of using **break** can be, for example, placing specific selection rules (such as rules located deeper in the tree) before the generic rules.
- In case the rule list contains specified attributes, and several rules use the same **<what>** variable, **break** must be added to the end of the rule as follows:

```
by * break
```

If **by *** is omitted from **break**, it will be likely unsuccessful, since **break** only applies to the previously mentioned user by default.

When configuring access control rules, consider the following:

- The list of authorized users can be defined in any rule of the list: there is no need to repeat the whole rule again. In case a rule is repeated, the system ignores subsequent appearances of the same rule, unless **break** is used in them.



- If a rule uses **regex**, optimize the search by inserting the caret (^) character at the beginning of the line, to indicate that the line begins with the given string.
- Empty ACLs provide read access to the DIT for any user.

4.3.2.2 Access Control Configuration Examples

This section offers some example access control lists. In the below examples, **dc=operator,dc=com** must be replaced with *cudbRootEntryDn* (as defined in *CUDB Node Configuration Data Model Description*, Reference [3]).

Note: Actual access control lists may be much longer than the examples shown below.

```
access to dn.regex="ou=admin, "
  by users auth
```

```
access to *
  by * write
```

Example 4 *Default ACL in CUDB, allowing write access for every user to the whole schema, except to the entries where users are defined:*

```
access to dn.children="ou=admin,dc=operator,dc=com"
  by users auth

access to dn.regex="serv=csps,mscId=([^\,]+),ou=multiSCs,dc=operator,dc=com"
  by dn.children="ou=hlrUsers,ou=admin,dc=operator,dc=com" write
  by dn.children="ou=pgUsers,ou=admin,dc=operator,dc=com" write

access to dn.regex="serv=ims,mscId=([^\,]+),ou=multiSCs,dc=operator,dc=com"
  by dn.children="ou=imsUsers,ou=admin,dc=operator,dc=com" read
  by dn.children="ou=pgUsers,ou=admin,dc=operator,dc=com" read
```

Example 5 *ACL taking advantage of LDAP groups, granting the following operations: pgUsers group members have write permission for serv=csps,mscId=* and serv=ims,mscId=*; hlrUsers group members have write permission for serv=csps,mscId=*; finally, imsUsers group members have read permission for serv=ims,mscId=.**



```

access to dn.children="ou=admin,dc=operator,dc=com"
  by users auth

access to dn.regex="serv=ims,mscId=([^\,]+),ou=multiSCs
,dc=operator,dc=com"
by dn.exact="cudbUser=hlr,ou=admin,dc=operator,dc=com" write
by dn.exact="cudbUser=pg,ou=admin,dc=operator,dc=com" write

access to dn.regex="serv=ims,mscId=([^\,]+),ou=multiSCs
,dc=operator,dc=com"
  by dn.exact="cudbUser=hss,ou=admin,dc=operator,dc=com" write
  by dn.exact="cudbUser=pg,ou=admin,dc=operator,dc=com" write

access to dn.regex="serv=ims,assocId=([^\,]+),ou=associations
,dc=operator,dc=com"
by dn.exact="cudbUser=hlr,ou=admin,dc=operator,dc=com" write
by dn.exact="cudbUser=pg,ou=admin,dc=operator,dc=com" write

access to dn.regex="serv=ims,assocId=([^\,]+),ou=associations
,dc=operator,dc=com"
  by dn.exact="cudbUser=hss,ou=admin,dc=operator,dc=com" write
  by dn.exact="cudbUser=pg,ou=admin,dc=operator,dc=com" write

access to dn.subtree="ou=identities,dc=operator,dc=com"
  by dn.exact="cudbUser=hlr,ou=admin,dc=operator,dc=com" read
  by dn.exact="cudbUser=hss,ou=admin,dc=operator,dc=com" read
  by dn.exact="cudbUser=pg,ou=admin,dc=operator,dc=com" write

access to dn.subtree="ou=servCommonData,dc=operator,dc=com"
  by dn.exact="cudbUser=hlr,ou=admin,dc=operator,dc=com" read
  by dn.exact="cudbUser=hss,ou=admin,dc=operator,dc=com" read
  by dn.exact="cudbUser=pg,ou=admin,dc=operator,dc=com" write

access to dn.subtree="ou=mscCommonData,dc=operator,dc=com"
  by dn.exact="cudbUser=hlr,ou=admin,dc=operator,dc=com" read
  by dn.exact="cudbUser=hss,ou=admin,dc=operator,dc=com" read
  by dn.exact="cudbUser=pg,ou=admin,dc=operator,dc=com" write

```

Example 6 *This complex example grants the following permissions: pgUsers group members have write permissions for everything, except the "ou=admin" branch; hlrUsers group members can read all the data, and have write permission for serv=ims,mscId=*, and serv=ims,assocId=*; finally, the hssUsers can read all the data, and have write permission for serv=ims,mscId=* and sev=ims,assocId=*.*

Note: ACLs can use both attributes and object classes, since object classes are essentially shorthands for the available attributes.

4.3.3 Storage of LDAP users passwords

CUDB makes it possible to store the passwords of the LDAP users in clear text or in hashed form. In case hashed form is selected, several types of hashes are allowed.

4.3.3.1 Default Storage of LDAP Users Passwords

Perform the following steps to configure the way passwords are stored for LDAP users:

1. On each node, configure the default form in which passwords for LDAP users are stored by modifying the `nodeLdapAuth` attribute located under the `CudbLdapAccess` class. For more information, refer to the



Class CudbLdapAccess section in *CUDB Node Configuration Data Model Description*, Reference [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).

The possible values for this attribute are "SASL" for clear-text storage and "SIMPLE" for hash form storage at user creation time.

2. On each node, configure the default node hash type by modifying the `nodeLdapHash` attribute located under the `CudbLdapAccess` class. For more information, refer to the *Class CudbLdapAccess* section in *CUDB Node Configuration Data Model Description*, Reference [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).

4.3.3.2

Storage of a Specific LDAP User Password

Perform the following steps to configure the way the password is stored for a specific LDAP user, overriding the default configuration:

1. Modify the following attributes:
 - Configure the value of the password storage form by setting the value of the `userLdapAuth` attribute located under the `CudbLdapUser` class.
 - Configure user hash type storing the password, in case of password stored in hash form, by setting the value of the `userLdapHash` attribute located under the `CudbLdapUser` class.
 - Configure user password by setting the value of the `cudbUserPassword` attribute located under the `CudbLdapUser` class.

Note: For more information on the `CudbLdapUser` class, refer to the *Class CudbLdapUser* section in *CUDB Node Configuration Data Model Description*, Reference [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).

2. Execute the `updateUserInfo` administrative operation on each node to align user information.



Refer to the *updateUserInfo* section in *CUDB Node Configuration Data Model Description*, Reference [3] for further information about the *updateUserInfo* administrative operation.

Note: The password of the `rootdn` can only be stored encrypted.

The user password is stored according to the values of `nodeLdapAuth` and `nodeLdapHash` (or `userLdapAuth` and `userLdapHash`, if defined) which are configured at the time the user password is set.

4.4 Switch User Management

When configuring the switch, take into account the considerations regarding user account management (see Section 4.1 on page 6) and secure password policies (see Section 4.1.3 on page 10). For CUDB systems deployed on native BSP 8100, SCXB security features can also be enforced..

4.4.1 Remote Switch User Authentication Management

For CUDB systems deployed on native BSP 8100, the procedure to configure the external user authentication/authorization is described in the *Configure LDAP Authentication* document in the BSP 8100 CPI.

For CUDB systems deployed on a cloud infrastructure, refer to the infrastructure documentation for information about user authentication management. Implementing such policies is recommended.

4.5 Router User Management

This section provides information on the proper user management on CMX routers for CUDB systems deployed on native BSP 8100.

CMX uses static accounts that cannot be changed. The following set of Linux users is available:

- Basic: This user provides basic read-only access.
- Advanced: This user provides limited read-write access.
- Root: This user provides super-user access.

Note: The root user is available only for Ericsson CMX support personnel.

The user names are hard-coded, and it is not possible to change them, or add new ones. For security reasons, the default passwords must be changed to secure CMX.

For CUDB systems deployed on a cloud infrastructure, refer to the infrastructure documentation for information about user authentication management. Implementing such policies is recommended.



5 System Audit Logging

This section describes how to perform a complete audit of the system for security purposes.

5.1 LDE Audit Logging

LDE authentication information is stored in the following file:

```
/var/log/<hostname>/auth
```

The file includes information on logging attempts and user creation operations.

Security events are collected to a separate dedicated log file stored at the following location:

```
/var/log/sec_events/security_events.log
```

For more information on security log events, refer to *CUDB Node Logging Events*, Reference [4].

5.2 Switch Audit Logging

For CUDB systems deployed on native BSP 8100, SCXB is configured to log events related to the following event groups:

- Security events
- Management actions
- Alarms
- Alerts
- Additional system information

Refer to the BSP 8100 CPI documentation for more information on switch logging.

For CUDB systems deployed on a cloud infrastructure, refer to the infrastructure documentation for information about user infrastructure audit logging.



5.3 Router Audit Logging

For CUDB systems deployed on native BSP 8100, consult the next level of maintenance support for information on the security management of security audit for CMX.

For CUDB systems deployed on a cloud infrastructure, refer to the infrastructure documentation for information about user infrastructure audit logging.

6 Certification Authority

The TLS configuration of the CA is centralized in the `CudbSystemSecurity` class. Configure the certificate list for CAs in case an application offers TLS capability for secure communication.

Note: CUDB supports SSL for backward compatibility reasons. However, using this protocol is heavily discouraged due to its known security breaches. Instead, only the latest TLS version is recommended.

The functions currently supporting TLS are as follows:

- LDAP access
- Database replication
- Notifications
- Remote authentication between CUDB nodes and an external authentication server using LDE.
- Sending logs to an external log server.

CUDB supports the storage of CA certificates (refer to the online resource of the *OpenSSL Project*, Reference [11] for more information).

Perform the following steps on each node to set the certificate list for CAs:

1. Store the file containing the certificate list for trusted CAs in a directory with persistent mount point, for instance in a subdirectory at the following location:

```
/home/cudb/security/config/certificates/ca/.
```

2. Once the file containing the list of certificates has been stored, modify the `tlsCaCertificatesFile` attribute located under the `CudbSystemSecurity` class to the name of the certification



authorities file, along with its full path. For more information, refer to the *Class CudbSystemSecurity* section in *CUDB Node Configuration Data Model Description*, Reference [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).

3. If the CA configuration is changed after any of the functions supporting TLS have been configured, follow the additional step below:
 - Restart the database cluster master replication servers by issuing the following command: `cudbManageStore -a -o masterreplicationrestart`.

Instructions on public certificates and private key deployment are provided in the sections describing the configuration of the available security features (Section 8 on page 31, Section 9 on page 32, Section 10 on page 34, Section 11 on page 37, and Section 12 on page 38).

Periodically, the file containing the list of certificates for trusted CAs might be updated for security reasons. If that happens, copy the updated file to a persistent file system, and configure the file name as described in the procedure above. When changing the CA file, use a different file name than the previous one.

Note: Certificates cannot be removed once added. However, certificates can be changed by adding a new certificate.

Attention!

Changes in the SSL/TLS configuration require the rolling restart of the notification processes, resulting in the loss of queued notifications together with the notifications generated during the process switchover. Applying SSL/TLS changes during low traffic hours is strongly recommended.

7

Certificate Generation

TLS certificates must be generated for all applications that offer TLS capability for secure communication. TLS certificates in general can be issued both by IP



address or by domain name. As detailed below, some CUDB applications use certificates issued by IP address, and others by domain name:

- The applications currently supporting TLS with certificates issued by the IP address are as follows:
 - Secure database replication. The IP address to use for this certificate is SITE_VIP. Refer to *CUDB Node Network Description*, Reference [1].
 - Secure notification delivery. The IP address to use for this certificate is FE_VIP. Refer to *CUDB Node Network Description*, Reference [1].
 - LDAP security operations. The different LDAP operations are handled by different CUDB IP addresses. The addresses to use during certificate generation to secure the corresponding channel are as follows:
 - FE_VIP for secure LDAP traffic.
 - PROVISIONING_VIP for secure LDAP provisioning traffic.
 - SITE_VIP for secure LDAP proxy traffic.
 - CUDB OAM Centralized Authentication System Support. The IP address to use for the server certificates is the IP of the server in the OAM network. The client (that is, the CUDB node) does not have its own certificates for this function: only Certification Authority Certificate (CAcert) is used on the server.
- The applications currently supporting TLS with certificates issued by domain name are as follows:
 - Centralized Security Event Logging. Server and client certificates must be generated and issued by a domain name. In case the optional `logServerName` configuration model attribute of the `CudbLogCertificates` class is set, it must contain the name used to generate the server certificate.

8 Configuring Secure Database Replication

Secure database replication is a system-wide procedure involving several nodes.

Database replication traffic can be secured between master and slave replication servers. Execute the following steps to activate SSL/TLS



(Secure Socket Layer / Transport Layer Security) master replication server authentication and traffic encryption.

1. Follow the instructions at Section 6 on page 29 to configure the list of recognized CAs.
2. Acquire a TLS certificate using the IP address specified in Section 7 on page 30 for this application.

This certificate is used by all master replication servers running on this CUDB node. Make sure the CA signing this certificate is included in the list of recognized CAs (see Section 6 on page 29 for more information).

3. Rename the file containing the TLS certificate for the master replication servers to the following:

```
server-cert.pem
```

4. Store `server-cert.pem` at the following location:

```
/home/cudb/security/config/certificates/mysql/
```

5. Rename the file containing the TLS key for the certificate above to the following:

```
server-key.pem
```

6. Store `server-key.pem` at the following location:

```
/home/cudb/security/config/keys/mysql/
```

7. Configure the slaves to initiate replication through the SSL tunnel at the next reconnection to the master. To do that, set the value of the `secureMySQLReplication` attribute located under the `CudbSystemSecurity` class to `true`. For more information, refer to the *Class CudbSystemSecurity* section in *CUDB Node Configuration Data Model Description*, Reference [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).

8. Repeat the operation for each CUDB node in the system.
9. When finished, run the following command to restart the master replication servers:

```
cudbManageStore -a -o masterreplicationrestart
```

Repeat the above steps for each CUDB node in the system. Slave replication servers reconnect to the master replication servers, establishing the secure channel in this way.



9 Configuring Secure Notifications

Configuring the secure notifications is a system-wide procedure involving several nodes.

The security configuration of notifications protects traffic confidentiality, and guarantees secure authentication between client and server. Execute the following steps to activate SSL/TLS traffic encryption, with client and server-side authentication.

Attention!

Changes in the SSL/TLS configuration require the rolling restart of the notification processes, resulting in the loss of queued notifications together with the notifications generated during the process switchover. Applying SSL/TLS changes during low traffic hours is strongly recommended.

9.1 TLS Configuration

Secure notifications require that the CA signing the SOAP server certificate is included in the recognized CA list. See Section 6 on page 29 for more information.

Typically, authentication scenarios are client-side: only the client authenticates the server. However, mutual authentication can also be configured: in such cases, the server also authenticates the client.

To configure secure notifications, provide a TLS key and the required certificates, then configure the file paths (no path is configured for these files by default). CA certificate is mandatory for the CUDB SOAP client to be able to check server identity. In case of mutual authentication, the server authenticates the client by providing a CUDB SOAP client public certificate and private key.

Perform the following steps to configure TLS authentication in CUDB:

1. Follow the instructions of Section 6 on page 29 to configure the list of recognized CAs.
2. Acquire a TLS certificate for the IP address specified in Section 7 on page 30 for this application.
3. Make sure that the CA signing this certificate is included in the list of recognized CAs. See Section 6 on page 29 for more information.



4. To configure mutual authentication, store the client public certificate and the private key to a directory with persistent mount point. For example, store the certificate at `/home/cudb/security/config/certificates/soap/` and store the key at `/home/cudb/security/config/keys/soap/`.
5. Follow the procedure below to configure mutual authentication:
 - Set the value of the `tlsCertificateFile` attribute located under the `CudbSoapCertificates` class to the name of the TLS certificate file, along with its full path. For more information, refer to the *Class CudbSoapCertificates* section in *CUDB Node Configuration Data Model Description*, Reference [3].
 - Set the value of the `tlsCertificateKeyFile` attribute located under the `CudbSoapCertificates` class to the name of the private key file, along with its full path. For more information, refer to the *Class CudbSoapCertificates* section in *CUDB Node Configuration Data Model Description*, Reference [3].
 - Set the value of the `URI` attribute located under the `CudbNotificationEndpoint` class to the Uniform Resource Identifier (URI) of a secure notification server using an https schema. For more information, refer to the *Class CudbNotificationEndPoint* section in *CUDB Node Configuration Data Model Description*, Reference [3].
 - Set the value of the `webService` attribute located under `CudbNotificationEndpoint` class to the path used to reach the web service. For more information, refer to the *Class CudbNotificationEndPoint* section in *CUDB Node Configuration Data Model Description*, Reference [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).

If client-side TLS certificates are to be changed for security reasons, repeat the procedure above and make sure to use different names for the TLS certificate file and the corresponding private key file than the ones used previously.

If notifications are running on secured connection, the TLS certificates and keys might be periodically updated for security reasons. If this happens, copy these files to a persistent file system, and configure the paths as described above. Change file names to apply the configuration.

Note: Certificates and keys cannot be removed once set. However, they can be changed by setting a new certificate or key.



10 Configuring LDAP Front End Security

Configuring LDAP FE security is a system-wide procedure involving several nodes.

The LDAP FE authentication and traffic confidentiality can be protected by configuring an SSL/TLS security layer for communications between client and server. In case SSL/TLS is not configured or not available, the SASL can be configured instead. This enables the client to send the hash of the password to the server during authentication, protecting password confidentiality in the network in this way.

10.1 LDAP Authentication Types

The LDAP FE supports both simple and SASL binds.

When using SASL bind authentication only the user name but not the user DN must be used.

The following restrictions apply to users requiring SASL binds:

- The user cannot belong to any LDAP group and the username cannot contain upper-case letters.
- The password of the user cannot be stored in hashed form (see Section 4.3.3 on page 25).
- SASL bind is not supported for the `rootdn` user.

10.2 Configuring TLS for LDAP Front End

To configure secure LDAP FE over TLS, a CA certificate, a public certificate, and a private key must be provided in a directory with persistent mount point. For example, certificates can be stored in `/home/cudb/security/config/certificates/ldapfe/`, while the keys can be copied to `/home/cudb/security/config/keys/ldapfe/`.

After the files have been copied, configure the file paths, as no default path is configured for these files.



Attention!

Changes in the SSL/TLS configuration require the rolling restart of the LDAP FE processes which closes the connections towards these processes, and also results in the loss of ongoing operations. In addition, LDAP operations may be rejected with error code 2 (protocol error) until the configuration changes are applied in all CUDB nodes. Applying SSL/TLS changes during low traffic hours is strongly recommended.

Perform the following steps to configure TLS for LDAP FE:

1. Follow the instructions of Section 6 on page 29 to configure the list of recognized CAs.
2. Acquire a TLS certificate using the IP address specified in Section 7 on page 30 for this application.
3. Make sure that the CA signing these certificates is included in the list of recognized CAs. See Section 6 on page 29 for more information.
4. Modify the following attributes on each node:
 - Set the value of the `tlsCertificateFile` attribute located under the `CudbLdapCertificates` class to the name of the TLS certificate file with its full path. For more information, refer to the *Class CudbLdapCertificates* section in *CUDB Node Configuration Data Model Description*, Reference [3].
 - Set the value of the `tlsCertificateKeyFile` attribute located under the `CudbLdapCertificates` class to the name of the private key file with its full path. For more information, refer to the *Class CudbLdapCertificates* section in *CUDB Node Configuration Data Model Description*, Reference [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).

If the LDAP server TLS certificates must be changed for security reasons, repeat the procedure above, but make sure to use different names both for the TLS certificate file and for the corresponding private key file than the ones used earlier.

Note: Certificates and keys cannot be removed once set. However, certificates and keys can be changed by setting a new certificate and key.



10.3 Configuring Secure LDAP Proxy

To configure the secure LDAP proxy, perform the following steps:

1. Provide the TLS certificates for every CUDB node in the system, as explained in Section 10.2 on page 35.
2. Make sure that the certificate has been generated by using the `SITE_VIP`, as described in Section 7 on page 30.

Note: Consider the special situations in case the IP addresses specified by the `cudbVIP` and `trafficVIP` are different.
3. To apply the secure proxy configuration, set, on each node, the value of the `secureLdapProxy` attribute located under the `CudbSystemSecurity` class to `true`. For more information, refer to the *Class CudbSystemSecurity* section in *CUDB Node Configuration Data Model Description*, Reference [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).

Note: Connections opened towards LDAP FE are lost during TLS proxy configuration, since the LDAP FEs are restarted to reload the configuration.

11 Configuring Secure CUDB OAM Centralized Authentication System Support

The **CUDB OAM Centralized Authentication System Support** function supports connection to remote LDAP authentication servers with TLS enabled or disabled. If TLS is enabled, the following TLS certificates are required:

- A private key file
- A public certificate file
- A CAcert file used to sign the above key and certificate files.

The function supports server side certificates only, and not client certificates. If TLS is set to enabled, the following server and client side requirements must be kept:

- The server must provide and set its own configuration files. In case of OpenLDAP, these files are as follows:



- `slapd.conf`
 - `CAcert`
 - The server private key
 - The server public certificate
- The client (that is, the CUDB node) requires only the same `CAcert` file that is used on the server, and not any client specific certificates.

The same `CAcert` file that is used to sign the server certificates must also be stored on each CUDB node, under `/cluster` to be available from all the blades.

The nodes must be configured to use `CAcert` by setting the `tlsCaCertificatesFile` attribute located under the `CudbSystemSecurity` class to the full path of the file containing a list of certificates for trusted Certificate Authorities (CAs) and CAs that signed the certificates stored in a CUDB node. For more information, refer to the *Class CudbSystemSecurity* section in *CUDB Node Configuration Data Model Description*, Reference [3].

Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3] for more information on all the steps required to modify the object model (for example, on using the `applyConfig` administrative operation to activate the changes).

As this model attribute is shared with notifications and LDAPs to set a `CAcert` for the CUDB node, the file can already exist in some cases, and may not contain the required new server `CAcert`. To solve this situation, add the new required `CAcert` content to that file, appending it to the previous content (the file can contain more than one CA). See Section 6 on page 29 for more information.

Because the client does not have its own certificates, remote LDAP authentication servers must be configured to not require client certificates. For example, in case of OpenLDAP remote servers, `TLSVerifyClient` can not be set to demand.

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



12 Configuring Secure Centralized Security Event Logging

The **Centralized Security Event Logging** function can be configured to send logs to log servers over TCP either with using TLS encryption, or without it. TLS is automatically enabled if the key, the public client certificates, and the CAcert are all set in the configuration model. These components are configured with the following configuration classes:

- The public certificate file is set with the `tlsCertificateFile` attribute of the `CudbLogCertificates` class.
- The private key is set with the `tlsCertificateKeyFile` attribute of the `CudbLogCertificates` class.
- The CAcert which is used to sign client certificates is set with the `tlsCaCertificatesFile` attribute of the `CudbSystemSecurity` class.

If any of these three attributes are not set, TLS will be automatically disabled, and logs will be sent without TLS encryption.

Enabling TLS on the client side requires the server to also have its own certificates set on its config file. The server also needs to have a private key and public certificate files that are different from the client ones, along with a CAcert. That CAcert, however, must be similar on the client (that is, on the CUDB node) and on the external log server, so that it can be used to sign all client and server certificates.

Note: CAcert is set with the `tlsCaCertificatesFile` attribute of the `CudbSystemSecurity` class, which is also used to set CAcert for other CUDB applications that also use TLS. The file can contain more than one CA: therefore, in case the attribute is already set to a CAcert file for other applications, add the new required CAcert content to that file by appending it to the previous content. See Section 6 on page 29 for more information.

An extra level of security can also be added to ensure that the CUDB node connects only to the proper log servers. By configuring the optional `logServerName` attribute of the `CudbLogCertificates` class with a name or a regular expression, the CUDB node connects and sends its logs to an external server only if the certificate of the server was created with the specified name or pattern (specified with the `Common Name` field when generated). By default, that attribute has a value of `"*"`, which means it does not filter any server name.



Attention!

Log files may contain relevant system information which may result in security breaches if disclosed. Ericsson accepts no liability if the encryption of the communication channel used to send the log files to a remote server is not correctly configured, or if these log files are not correctly protected against improper access or use when stored in the remote server, and therefore the information within the log files is used with fraudulent purposes.

12.1 Examples of Using Secure Centralized Security Event Logging

This section provides examples on using **Centralized Security Event Logging** with and without using TLS.

12.1.1 Using TLS

The function is set up with the following example steps:

1. Configure and start a remote log server that supports TLS. In the below example, `rsyslog v5` is used, configured to listen on TCP port 2999 and to store all received logs into the `/etc/testingTLS` local file. The configuration file that the server uses in this case is located in `/etc/rsyslog.conf`, while its contents are as shown in Example 7.
2. Once the server is up and running, configure the CUDB node to act as client and send its logs to the server mentioned in the previous step. To do so, set the following configuration model attributes to connect to that server:
 - Set the `externalLogServerIp` attribute of the `CudbExternalLogServer` class to configure the IP of the server in the OAM network.
 - Set the `externalLogServerPort` attribute of the `CudbExternalLogServer` class to 2999, as the server is listening on that port.
 - Set the `tlsCertificateKeyFile` attribute of the `CudbLogCertificates` class to configure the client certificate key file.
 - Set the `tlsCertificateFile` attribute of the `CudbLogCertificates` class to configure the client public certificate.
 - Include the CAcert used to sign the client and server certificates in the file indicated in the `tlsCaCertificatesFile` attribute of the `CudbSystemSecurity` class.
3. Finally, set the `enabled` attribute of the `CudbExternalLogMgmt` class to `true`.



After the above steps, logs will be sent to the external server encrypted with TLS. Refer to *CUDB Node Configuration Data Model Description, Reference [3]* for more information on the affected configuration objects and attributes listed above.

```
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
$DefaultNetstreamDriver gtls

$DefaultNetstreamDriverCAFile /etc/tlsCerts/ca.pem           # CAcert file
$DefaultNetstreamDriverCertFile /etc/tlsCerts/server-cert.pem # server public certificate file
$DefaultNetstreamDriverKeyFile /etc/tlsCerts/server-key.pem  # server private key certificate file

$ModLoad imtcp
$InputTCPServerStreamDriverMode 1 # run driver in TLS-only mode
$InputTCPServerStreamDriverAuthMode x509/name # authenticate by hostname
$InputTCPServerStreamDriverPermittedPeer * # accept connections of clients with certificates generated with a
$InputTCPServerRun 2999 # listen on tcp port 2999

*. * /etc/testingTLS # destination of received logs
```

Example 7 Example of rsyslog Configuration File on the Remote Server

Note: The `InputTCPServerStreamDriverPermittedPeer` parameter is set to "*" on this example. This means that the clients can have certificates generated with any name. If the parameter is set to a regular expression containing a name, then it establishes connections and accepts logs only from clients that have certificates generated with a Common Name which matches that regular expression.

For example, if set to `CUDBLogClient*`, a client with a certificate-created setting Common Name of `CUDBLogClient1` will be allowed, while others not matching the specified pattern will not. If this extra level of security is not needed to filter log sources, then set the parameter to "*" as shown above.

This setting is similar to the `logServerName` attribute of the `CudbLogCertificates` class, as described in Section 7 on page 30. The difference is that this is a server-side setting instead of a client side configuration, so it filters clients instead of servers.

In this example, the log server is started with following parameters:

```
rsyslog -c 5 -n -f /etc/rsyslog.conf
```

12.1.2 Not Using TLS

The function is set up with the following example steps:

1. Configure and start a remote log server. In the below example, `rsyslog v5` is used, configured to listen on TCP port 2999 and to store all received logs into the `/etc/testingRemoteLogging` local file. The configuration file that the server uses in this case is located in `/etc/rsyslog.conf`, while its contents are as shown in Example 8.



2. Once the server is up and running, configure the CUDB node to act as client and send its logs to the server mentioned in the previous step. To do so, set the following configuration model attributes to connect to that server:
 - Set the `externalLogServerIp` attribute of the `CudbExternalLogServer` class to configure the IP of the server in the OAM network.
 - Set the `externalLogServerPort` attribute of the `CudbExternalLogServer` class to 2999, as the server is listening on that port.
3. Finally, set the `enabled` attribute of the `CudbExternalLogMgmt` class to `true`.

After the above steps, logs will be sent to the external server. Refer to *CUDB Node Configuration Data Model Description*, Reference [3] for more information on the affected configuration objects and attributes listed above.

```
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
$ModLoad imtcp
$InputTCPServerRun 2999 # listen on tcp port 2999
*. * /etc/testingRemoteLogging # destination of received logs
```

Example 8 Example of rsyslog Configuration File on the Remote Server

In this example, the log server is started with following parameters:

```
rsyslog -c 5 -n -f /etc/rsyslog.conf
```

13 Other Security Recommendations

Besides the security features outlined in this document, It is strongly advised to also observe the following security recommendations:

- Avoid executing OAM actions with the root user whenever possible. OAM operations must be performed with the `cudbadmin` user, which is automatically created during system installation.
- Avoid console and bash history LDAP password persistency by performing LDAP binds using the following command:

```
-y <passwordfile>
```

In the above example, `<passwordfile>` is a file storing the password for LDAP. Remove this file if it is not used anymore, or before the session is closed.



Alternatively, the `-w` option can be used, forcing a prompt requiring the LDAP password (and therefore leaving no persistency neither in the console, nor in the bash history).

- It is not recommended to use password authentication for the root account for security reasons: use certificate-based authentication instead.

Password authentication of the root account can be disabled by setting the `PermitRootLogin` configuration parameter to the `without-password` value in the configuration files `/home/cudb/security/config/sshd_config_sc` and `/home/cudb/security/config/sshd_config_pl` for the SCs and payload blades or VMs, respectively. These modifications must be persistent to reboot. To ensure persistency to reboot, use the `LDE etc-overlay-framework` as described in *LDE Management Guide*, Reference [8].

- It is extremely important to secure the physical access to the infrastructure equipment. Such measures include locking the cabinet doors, posting security personnel to guard the infrastructure, installing video surveillance systems recording footage, and other security measures which ensure that no strangers can access the infrastructure equipment. If intruders have physical access to the servers, they own them. In case of local access to the system, so many attack possibilities are available to intruders that it is almost impossible to avoid them all without greatly affecting server performance.
- A list of scheduled maintenance actions are available for execution to maintain system integrity and security. Refer to *CUDB Node Preventive Maintenance*, Reference [5] for more information on these actions.
- For CUDB systems deployed on a cloud infrastructure, the cloud infrastructure must fulfill the security requirements detailed in *CUDB Virtual Infrastructure Requirements*, Reference [6].

14 Privacy

This section contains privacy-related statements regarding the CUDB system.

14.1 Notice

CUDB may store personal information, and may impact the right to privacy of the data subjects (that is, subscribers) whose data is stored. The specific data items to be stored depend on the application(s) using the CUDB system.



When operating the CUDB system as a data controller, ensure that personal information is stored in a fair and lawful manner, and in accordance to the local data protection regulation in effect. This can be achieved by providing notice to the subscribers about the privacy policies of the operator, for example at the moment of establishing the subscription.

It is also advised to provide comprehensive and understandable information to subscribers prior to, or at the time of collecting the personal information.

14.2 Consent

CUDB may also store sensitive personal data in addition to basic personal data. The local data protection regulations where CUDB is operated may require obtaining subscriber consent to process this kind of personal information. Such consent must be obtained to allow the following activities:

- Collecting and maintaining personal data of the subscriber, aimed at holding securely this information.
- Fulfilling the purpose of installing, upgrading, and administrating the CUDB system.
- Disclosing personal information to third parties.



Glossary

For the terms, definitions, acronyms and abbreviations used in this document, refer to *CUDB Glossary of Terms and Acronyms*, Reference [7].





Reference List

CUDB Documents

- [1] *CUDB Node Network Description*
- [2] *CUDB Users and Passwords*, 3/006 51-HDA 104 03/10
- [3] *CUDB Node Configuration Data Model Description*
- [4] *CUDB Node Logging Events*
- [5] *CUDB Node Preventive Maintenance*
- [6] *CUDB Virtual Infrastructure Requirements*
- [7] *CUDB Glossary of Terms and Acronyms*

Other Ericsson Documents

- [8] *LDE Management Guide*
- [9] *ESA Performance Management*

Other Documents and Online References

- [10] *OpenLDAP Software 2.4 Administrator's Guide*
- [11] *OpenSSL Project* <http://www.openssl.org/>
- [12] *An Approach for Using LDAP as a Network Information Service. IETF RFC 2307* <https://tools.ietf.org/html/rfc2307>