ERICSSON

CUDB System Administrator Guide

SYS ADM GUIDE

2/1543-HDA 104 03/10-V1 Uen A

Contents

1 Introduction

This document describes every system-level, node-level and auxiliary administration and configuration procedure available for the Ericsson Centralized User Database (CUDB).

1.1 Document Purpose and Scope

The purpose of this document is to provide detailed information about the administration and configuration procedures available for CUDB.

Therefore, the guide describes the following:

- The resources and settings available to access and configuration in a CUDB node.
- The interfaces and tools through which the above resources and settings can be accessed and configured.
- The steps of performing management procedures.

The final execution permissions on a procedure are determined by taking into consideration the more restrictive permissions applied to every command executed during the procedure. Refer to *CUDB Node Commands and Parameters*, Reference [1] for user type permissions.

1.2 Revision Information

Rev. A

This document is based on 2/1543-HDA 104 03/9 with the following changes:

- Removed obsolete hardware information throughout the document.
- Virtualization terminology updates throughout the document.
- “OAM Automation with NETCONF Support” updates throughout the document.
- Section 3.2.2 on page 5: Added command for `cudbadmin` user.
- Section 16.2.2 on page 16: New section on using the Atlas GUI and OpenStack command-line tools to perform node startup after a graceful shutdown in case of using CEE 16A.

- Section 16.2.2.1 on page 16: New section on starting a node after graceful shutdown using the Atlas GUI.
- Section 16.2.2.2 on page 18: New section on starting up a node after graceful shutdown using the OpenStack command-line tools.
- Section 16.3 on page 20: Updated to add section reference.
- Section 16.3.1 on page 20 and Section 16.3.2 on page 20: New sections on starting up a node after a non-graceful shutdown.
- Section 16.5 on page 21: Added information and references regarding the Atlas GUI and OpenStack command-line tools.
- Section 16.5.2 on page 23: New section on using the Atlas GUI and OpenStack command-line tools to perform a graceful shutdown in case of using CEE 16A.
- Section 16.5.2.1 on page 23: New section on gracefully shutting down a node using the Atlas GUI.
- Section 16.5.2.2 on page 24: New section on gracefully shutting down a node using the OpenStack command-line tools.
- Section 18.1 on page 25: Updated procedure of configuring eVIP routing.
- Section 18.2 on page 25: Updated Step 4.
- Section 18.3 on page 26: Updated Step 3.
- Section 18.4 on page 26: Updated note to include information on LDAP FE node and server counters. Added a note about the creation of new DSG(s).
- Section 18.24 on page 33 and Section 20.15 on page 42: Added information on requirements.
- Section 19.5.2 on page 36: Updated command with 'sudo'.
- Section 19.7 on page 37: Updated command with 'sudo'.
- Section 19.9 on page 37: Renamed section.

- Section 20.26 on page 45: Updated the procedure for updating CUDB LDAP user information in a CUDB node.
- Section 20.30 on page 47: Renamed and completely updated the section. Added a reference to both sections on the Object Model Modification Procedure. Updated procedure with an extra step and also the example.
- Section 20.31 on page 48: Renamed and completely updated the section.
- Section 20.33 on page 51: Updated commands and examples with `sudo`.

1.3 Typographic Conventions

Typographic conventions can be found in the following document:

- *Typographic Conventions*

2 Overview

CUDB provides several interfaces and procedures for configuring CUDB nodes. All CUDB nodes in a CUDB system are managed and configured in the same way.

The available interfaces and configuration procedures are described in the sections below.

3 Interfaces and Tools

This section provides a description of the interfaces and tools used for CUDB system administration.

3.1 Interfaces

This section describes every Operation and Maintenance (OAM) interface used to access and manage CUDB. An overall view of these interfaces is shown in Figure 1. More information on each interface is available in the following chapters.

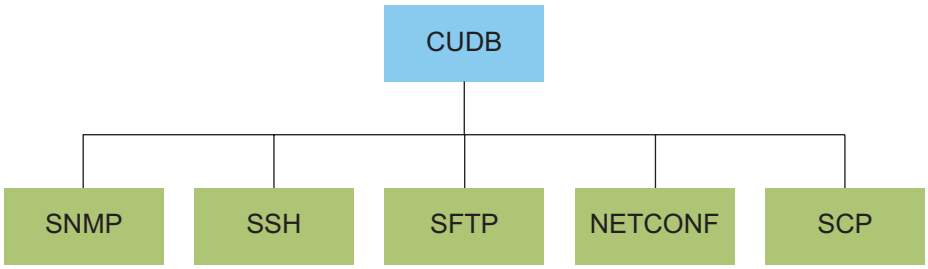

Figure 1 OAM Interfaces

3.1.1 SNMP

The Simple Network Management Protocol (SNMP) v3 is a standard protocol used to interchange administrative information between network elements using the User Datagram Protocol (UDP). Fault reporting in CUDB happens through trap messages sent through SNMP: the trap messages send alarm information to the Network Management System (NMS).

3.1.2 SSH

Secure Shell (SSH) is a network protocol allowing data exchange through a secure channel between two network devices.

SSH is used in CUDB to provide access to the Command Line Interface (CLI).

3.1.3 SFTP

The Secure File Transfer Protocol (SFTP) is built on top of SSH, and is used to remotely and securely access files stored in a CUDB node.

3.1.4 NETCONF

The Network Configuration Protocol (NETCONF) is built on top of SSH, and is a machine-to-machine interface providing means to install, modify, and delete the configuration of the network devices. The CUDB NETCONF interface is provided by Common Operation and Maintenance (COM) component and can

be used for configuration purposes instead of the CUDB Configuration CLI (see Section 3.2.2 on page 5).

The CUDB NETCONF interface provided by COM only supports some of the capabilities defined in the standards.

CUDB NETCONF clients must be configured to use a `commit at <close-session>` commit behavior. Refer to *COM Management Guide*, Reference [23] and *COM NETCONF Interface Base*, Reference [24] for further details about NETCONF.

3.1.5 SCP

The Secure Copy Protocol (SCP) has the same purpose and use as of SFTP but implements a more efficient transfer algorithm.

3.2 Tools

This section describes the tools used to access and configure CUDB.

3.2.1 CUDB CLI

CUDB provides a CLI interface to execute both Linux Distribution Extension (LDE) and specific CUDB commands (refer to *LDE Management Guide*, Reference [25] and *CUDB Node Commands and Parameters*, Reference [1]). Access to the CLI interface of the CUDB nodes is offered through SSH using port 22.

To access this CLI from the administration node, use the following command:

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

Connection to this CLI requires authentication. Refer to *CUDB Users and Passwords*, Reference [2] for user credentials that can be used to access the CUDB CLI.

3.2.1.1 Finding the Active System Controller

Several troubleshooting resources (such as the `cudbGetLogs` and `cudbAnalyser` scripts, or the CoreMW console) can be executed only on the active System Controller (SC). If needed, use the following command to determine which is the active SC:

```
# cudbHaState | grep COM | grep ACTIVE
```

The expected output must be similar to the below example:

```
COM is assigned as ACTIVE in controller SC-1.
```

3.2.2 CUDB Configuration CLI

CUDB provides a CLI interface to perform configuration updates. This CLI is based on the COM CLI and can be reached from a regular CUDB CLI session on the active SC. Refer to *COM Management Guide*, Reference [23] for more information.

To access the configuration CLI from a regular CLI session, follow the steps below:

1. Establish a CUDB CLI session towards the CUDB node, use the following command:

   ```
   ssh -l cudbadmin <CUDB_Node_OAM_IP_Address>
   ```

2. Use the command below (the output is also shown) to find the active SC:

   ```
   cudbHaState | grep COM | grep ACTIVE

   COM is assigned as ACTIVE in controller SC-1
   ```

   The active SC (SC_2_1 in the example above) must be used for accessing the COM CLI.

   For `cudbadmin` user:

   ```
   sudo cudbHaState | grep COM | grep ACTIVE
   ```

3. Access the COM CLI by executing the following command:

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

3.2.3 CUDB Platform Interfaces

The following platform elements provide a CLI for configuration purposes:

- evolved Virtual IP (eVIP): Refer to *eVIP Management Guide*, Reference [26] for information about this CLI.
- Blade Server Platform 8100 (BSP 8100).

3.2.4 LDAP Schema Management Tools

CUDB provides tools for managing the Lightweight Directory Access Protocol (LDAP) schema, which can be executed on any Linux machine with Java, or also on the SCs of a CUDB node through SSH. The available tools are as follows:

- Schema Management Graphical User Interface (GUI): see Section 15.3 on page 12 for more information.
- Schema Update Tool: see Section 15.4 on page 12 for more information.

3.2.5 UDC Cockpit

UDC Cockpit tool, formerly known as CUDB Observability Tool is a GUI that is used to monitor the system at real-time, and to analyze historical data. The GUI monitors LDAP counters, mastership, and alarms. The Cockpit GUI is accessed through a web interface, and uses SSH and SNMP interfaces to communicate with the CUDB system.

4 Configuration Data Management

Most of the CUDB configuration data can be managed through the configuration model. Refer to *CUDB Node Configuration Data Model Description*, Reference [3] for more information.

Configuration data management through the configuration model is possible through the CUDB Configuration CLI and NETCONF interface. See Section 3.2.2 on page 5 for more information on how to access the CUDB Configuration CLI and Section 3.1.4 on page 4 for more information on the NETCONF interface.

5 Fault Management

Fault management in CUDB provides a set of services that perform the following roles:

- Notify the system administrator if a problem is detected that requires human intervention.
- Notify the system administrator if particular events occur.

For more information about fault management, refer to *CUDB Node Fault Management Configuration Guide*, Reference [4].

6 Performance Management

Performance management in CUDB provides two types of statistics:

- CUDB performance indicators, which contain information about CUDB behavior. These counters provide performance information for each CUDB node. Refer to *CUDB Counters List*, Reference [5] for further information.
- Application statistics, which provide information related to the application data stored in the CUDB system. The statistics are collected and provided on system level. Refer to *CUDB Performance Guide*, Reference [6] for further information.

7 CPU Load Monitoring

CUDB can monitor the CPU load on all blades or Virtual Machines (VMs) of a CUDB node, and can present the monitoring results in a tabular format. Get information about CPU load monitoring with the `cudbMpstat` command as follows:

1. Establish a new administrative CUDB CLI session towards the target CUDB node with the following command:

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

2. Execute **`cudbMpstat`**.

3. Exit the script by pressing **Ctrl+C**.

Refer to *CUDB Node Commands and Parameters*, Reference [1] for further information.

8 Logging Management

Logging management in CUDB is based on the reported log events.

For further information about logging events in CUDB, refer to *CUDB Node Logging Events*, Reference [7]. Information on configuring the **Centralized Security Event Logging** function is also available in that document.

9 CUDB Log Collection

The CUDB system can collect the log files of a specific CUDB node or all the CUDB nodes with the `cudbCollectInfo` command.

This command collects the log files, and compresses them into `tar` file format.

For more information about this command, refer to *CUDB Node Commands and Parameters*, Reference [1].

10 Security Management

CUDB security management provides a set of features to ensure secure authentication and confidentiality for all administrative tasks and traffic scenarios.

Refer to *CUDB Security and Privacy Management*, Reference [8] for information about security management in CUDB.

11 Backup and Restore Procedures

CUDB offers three types of backup operations:

- **Data backup**: This operation creates data backups of the information stored in the database.
- **Software and configuration backup**: This operation creates data backups of the following information:
  - CUDB node middleware configuration.

  - CUDB node configuration.
  - Software packages installed on the CUDB node.
  - Auxiliary information related to CUDB and stored in the Processing Layer Database (PLDB).

- **Configuration backup of specific infrastructure element(s)**.

Refer to *CUDB Backup and Restore Procedures*, Reference [9] and *CUDB Node Preventive Maintenance*, Reference [10] for information on how to perform backup and restore operations in CUDB.

12 Replication

Replication is automatically handled by CUDB and requires no administrative actions other than handling raised alarms. In case of replication issues, follow the procedures described in the Operating Instructions (OPIs) for replication alarms (refer to *CUDB Node Fault Management Configuration Guide*, Reference [4] for further information).

13 Reconciliation

Reconciliation is automatically handled by CUDB and requires no administrative actions other than handling raised alarms. In case of reconciliation issues, follow the procedures described in the OPIs for reconciliation alarms (refer to *CUDB Node Fault Management Configuration Guide*, Reference [4] for further information).

Note: Handling reconciliation issues for the Subscription Reallocation procedure is an exception: in special cases, reconciliation must be requested manually. See Section 18.14 on page 29 for more information on Subscription Reallocation, and Section 18.15 on page 30 for more information on manual reconciliation.

Refer to *CUDB Data Storage Handling*, Reference [11] for more information on reconciliation in CUDB.

14 Networking

The general network infrastructure of the CUDB system is described in *CUDB Node Network Description*, Reference [12].

15 Data and Schema Management

This section provides information on the administrative tasks related to data and schema management.

15.1 LDAP Data Import and Export

The CUDB system supports the import of LDAP data from LDAP Data Interchange Format (LDIF) files to the database, and also the export of data from the database to LDIF files. Refer to *CUDB Import and Export Procedures*, Reference [13] for further information.

Import and export can also be performed by LDAP interface. This is commonly used for smaller amounts of data. Refer to Section 15.2 on page 11 for more information.

15.2 Adding and Searching LDAP Data

To perform addition or search of data, stored in CUDB, using the LDAP interface, standard LDAP clients with access to CUDB traffic network can be used.

The following chapters contain information about Open LDAP tools `ldapadd` and `ldapsearch`, which are delivered with CUDB, to perform these operations.

15.2.1 Adding LDAP Data

The following steps give an example of adding LDAP data from LDIF files using `ldapadd` command, from within the CUDB node:

1. Log on to the CUDB node with the PLDB master replica.
2. Transfer all LDIF files to the CUDB SC node.

Note: Verify the contents of each LDIF file and replace *<root_DN>* with the operator-specific root Distinguished Name (DN).

3. Import LDIF files using the `ldapadd` command, for example:

```
# ldapadd -x -c -H ldap://PL0:389 -D "cn=manager,<root_DN>" -w <manager_password> -S /tmp/error.log -f <path_to_ldif_file>
```

Note: Replace *<root_DN>* with the proper root DN.

For more information about LDAP root password, *<manager_password>* refer to *CUDB Users and Passwords*, Reference [2].

4. Verify that all `ldapadd` operations are successful by querying the corresponding `error.log` file for errors.

Note: Errors are related to common entries, for example, `dn: serv=CSPS, ou=mscCommonData, dc=`*<root_DN>* in multiple LDIF files can be ignored.

15.2.2 Searching LDAP Data

Searching LDAP data can be done by using `ldapsearch` command.

Refer to *OpenLDAP, ldapsearch Tool Manual*, Reference [27] for more information about the command.

15.3 LDAP Schema Edition

The Schema Tools software provides an easy-to-use graphical interface to create and edit LDAP schemas. For further information, refer to *CUDB LDAP Schema Management Graphical User Interface*, Reference [14].

Note: The graphical tool requires a Linux machine with Java.

For further information, refer to *CUDB LDAP Schema Management Graphical User Interface*, Reference [14].

Note: LDAP schema management is restricted to Ericsson personnel. Contact the next level of maintenance support to perform such procedures.

15.4 Schema Update and Index Creation on New Attributes

CUDB provides methods for changing LDAP application schemas by adding new object classes, or adding new attributes to already existing object classes. It is also possible to define indexes on new attributes, no matter if they belong to a new or an already existing object class. CUDB also provides methods to add new application schemas.

Note: LDAP schema update and index creation are restricted to Ericsson personnel. Contact the next level of maintenance support to perform such procedures.

15.4.1 Index Creation

New indexes in a local CUDB node can be created by setting the multivalued attribute `ldapAttrIndexes` of the `CudbLdapAccess` class in the configuration model. For more information, refer to the "Class CudbLdapAccess" section of *CUDB Node Configuration Data Model Description*, Reference [3].

Values of the `ldapAttrIndexes` attribute must contain all attributes that are to be indexed separated by commas.

Warning!

`LdapAttrIndexes` must be defined in the same order on every node.

Refer to 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 administrative operation `applyConfig` to activate the changes).

16 Startup and Shutdown

This section describes the startup and shutdown operations available in CUDB.

Note: System startup and shutdown must be performed by authorized Ericsson personnel only.

16.1 Initial System Startup

Note: Initial system startup must be performed by authorized Ericsson personnel only.

16.2 Node Startup after Graceful Shutdown

In general, starting a gracefully shut down CUDB node requires no special procedures other than powering it on, and handling the alarms that may be raised (and remain raised) after the node is back online.

16.2.1 Node Startup after a Graceful Shutdown in Case the CUDB System Is Deployed on Native BSP 8100 Hardware

The standard procedure for starting up a node in case the CUDB system is deployed on native BSP 8100 hardware is as follows:

1. Login to the BSP 8100 CLI.

2. Use the CLI console to check which shelf and slot numbers are associated with the SCs and payload blades. Execute the following commands to do so:

   ```
   >configure

   (config)>ManagedElement=1,DmxcFunction=1,Eqm=1,Virtua
   lEquipment=cudb

   (config-VirtualEquipment=cudb)>show-table -m Blade -p
   bladeId, userLabel,administrativeState
   ```

   The expected output must be similar to the below example:

   ```
   ============================================
   | bladeId | userLabel | administrativeState |
   ============================================
   | 0-1     | SC-1      | LOCKED              |
   | 0-11    | PL-6      | LOCKED              |
   | 0-13    | PL-7      | LOCKED              |
   | 0-15    | PL-8      | LOCKED              |
   | 0-17    | PL-9      | LOCKED              |
   | 0-19    | PL-10     | LOCKED              |
   | 0-21    | PL-11     | LOCKED              |
   | 0-23    | PL-12     | LOCKED              |
   | 0-3     | SC-2      | LOCKED              |
   | 0-5     | PL-3      | LOCKED              |
   | 0-7     | PL-4      | LOCKED              |
   | 0-9     | PL-5      | LOCKED              |
   ============================================
   ```

3. Use the CLI console to power on the SCs and payload blades fetched in the previous step. After the SCs are unlocked, continue with the payload blades:

   ```
   (config)>ManagedElement=1,DmxcFunction=1,Eqm=1,Virtua
   lEquipment=cudb

   (config-VirtualEquipment=cudb)>Blade=0-1,administrativ
   eState=UNLOCKED
   ```

```
(config-VirtualEquipment=cudb)>commit -s

(config-VirtualEquipment=cudb)>Blade=0-3,administrativ
eState=UNLOCKED

(config-VirtualEquipment=cudb)>commit -s

...

(config-VirtualEquipment=cudb)>end

(VirtualEquipment=cudb)>exit
```

4. Exit the CLI console.
5. Wait until the SCs are completely powered up.
6. After a few minutes, it is possible to log in to the SCs of the CUDB and verify that all blades in the CUDB are starting up. To do so, execute the following command:

   ```
   tipc-config -n
   ```

   Note: Blades that are up and running are indicated with `up` (see example below). Therefore, check that all lines contain `up`. If not all blades are running (indicated with `down`), wait approximately 10 minutes until all of them are initialized. Some blades which are not running can be missing from the list, so it must be checked that all blades are on the list and all of them are up.

   ```
   CUDB_79 SC_2_2# tipc-config -n
   Neighbors:
   <1.1.1>: up
   <1.1.3>: up
   <1.1.4>: up
   <1.1.5>: up
   <1.1.6>: up
   <1.1.7>: down
   <1.1.8>: up
   <1.1.9>: up
   <1.1.10>: up
   <1.1.11>: up
   <1.1.12>: up
   <1.1.13>: up
   <1.1.14>: up
   ```

   Contact the next level of maintenance support if one or more blades fail to initialize in 10 minutes.
7. Start the database cluster processes with the following command:

   ```
   sudo cudbManageStore -a -o start
   ```

Note: If the node has more than one subrack, the subrack with SCs and payload blades must be powered first.

16.2.2 Node Startup after Graceful Shutdown in Case the CUDB System Is Deployed on a Cloud Infrastructure

In case of using Cloud Execution Environment (CEE), starting a gracefully shut down CUDB node deployed on a cloud infrastructure can be done through the Atlas GUI or the OpenStack command-line clients. These procedures are described in Section 16.2.2.1 on page 16 and Section 16.2.2.2 on page 18, respectively. For more information about the Atlas GUI, refer to the ‘‘Atlas Dashboard End User Guide’’ document and for more information about the OpenStack command-line tools, refer to the ‘‘OpenStack End User Guide’’ document in the CEE CPI.

In case of using a different cloud solution, refer to the solution-specific documentation for more information.

16.2.2.1 Node Startup after Graceful Shutdown Using the Atlas GUI

To start a gracefully shut down CUDB node through the Atlas GUI, perform the following steps:

1. Log in to the Atlas Dashboard.
2. Select the appropriate project in the **Current Project** field and select **Project** in the **View** field.
3. Click the **Instances** category.
4. Mark both SCs of the CUDB and choose the **Start Instances** action located in the top bar, under the **More Actions** drop-down menu.
5. Verify that the status of both SCs is `Active` and the power state is `Running`.
6. After a few minutes, verify that both SCs are up and running. To do so, log in to the SCs and execute the following commands:

    a. **`tipc-config -n`**

    Note: VMs that are up and running are indicated with `up` (see example below). If one of the SCs is not running (indicated with `down`), wait a few minutes until it is up. VMs which are not running can be missing from the list, so check that both VMs are on the list and both of them are up.

```
CUDB_79 SC_2_1# tipc-config -n

Neighbors:
<1.1.2>: up
```

b. **cudbHaState**

Note: If both SCs are fully initialized, only payload VMs will be listed having problems in the output of the command under the ‘‘SU states’’ section. If the ‘‘SU states’’ section lists one of the SCs, it means that the mentioned VM is not fully initialized. If this happens, wait a few minutes, then execute the command again and see if the output has changed and that the SC is not listed anymore.

7. Mark all payload VMs of the CUDB and choose the **Start Instances** action located in the top bar, under the **More Actions** drop-down menu.

8. Verify that the status of all payload VMs is `Active` and the power state is `Running`.

9. After a few minutes, verify that all VMs are up and running. To do so, log in to the SCs and execute the following commands:

   a. **tipc-config -n**

   Note: VMs that are up and running are indicated with `up` (see example below). Therefore, check that all lines contain `up`. If not all VMs are running (indicated with `down`), wait approximately 15 minutes until all of them are initialized. VMs which are not running can be missing from the list, so check that all VMs are on the list and all of them are up.

   ```
   CUDB_79 SC_2_2# tipc-config -n

   Neighbors:
   <1.1.1>: up
   <1.1.3>: up
   <1.1.4>: up
   <1.1.5>: up
   <1.1.6>: up
   <1.1.7>: down
   <1.1.8>: up
   <1.1.9>: up
   <1.1.10>: up
   <1.1.11>: up
   <1.1.12>: up
   <1.1.13>: up
   <1.1.14>: up
   ```

   b. **cudbHaState**

Note: If all VMs are fully initialized, then the message "`Status OK`" will be listed in the output of the command under the "SU states" section. If some of the VMs are listed in this section, it means the mentioned VM is not fully initialized. In case this happens, wait a few minutes, then execute the command again and see if the status is changed and the VM is not listed anymore.

Contact the next level of maintenance support if one or more VMs fail to initialize in 15 minutes.

10. Start the database cluster processes with the following command:

    sudo cudbManageStore -a -o start

16.2.2.2 Node Startup after Graceful Shutdown Using OpenStack Command-Line Tools

To start a gracefully shut down CUDB node through the OpenStack command-line, execute the following commands from the Cloud Infrastructure Controller (CIC):

1. Execute the following command to start the SCs of the CUDB:

   nova start *<SC_id>*

   Note: Execute the command for both SCs.

2. Verify that the status of both SCs is `Active` and the power state is "1" (Running) with the following command:

   nova show *<SC_id>*

   Note: Execute the command for both SCs.

3. After a few minutes, verify that both SCs are up. To do so, log in to the SCs and execute the following commands:

   a. **tipc-config -n**

      Note: VMs that are up and running are indicated with `up` (see example below). If one of the SCs is not running (indicated with `down`), wait a few minutes until it is up. VMs which are not running can be missing from the list, so check that both VMs are on the list.

      ```
      CUDB_79 SC_2_1# tipc-config -n

      Neighbors:
      <1.1.2>: up
      ```

   b. **cudbHaState**

Note: If both SCs are fully initialized, only payload VMs will be listed having problems in the output of the command under the "SU states" section. If the "SU states" section lists one of the SCs, it means that the mentioned VM is not fully initialized. If this happens, wait a few minutes, then execute the command again and see if the output has changed and that the SC is not listed anymore.

4. Execute the following command to start payload VMs:

   nova start *<payload_vm_id>*

   Note: Execute the command for all payload VMs.

5. Verify that the status of payload VMs is `Active` and the power state is "1" (Running) with the following command:

   nova show *<payload_vm_id>*

   Note: Execute the command for all payload VMs.

6. After a few minutes, verify that all VMs are up and running. To do so, log in to the SCs and execute the following commands:

   a. **tipc-config -n**

      Note: VMs that are up and running are indicated with `up` (see example below). Therefore, check that all lines contain `up`. If not all VMs are running (indicated with `down`), wait approximately 15 minutes until all of them are initialized. VMs which are not running can be missing from the list, so check that all VMs are on the list and all of them are up.

      ```
      CUDB_79 SC_2_2# tipc-config -n

      Neighbors:
      <1.1.1>: up
      <1.1.3>: up
      <1.1.4>: up
      <1.1.5>: up
      <1.1.6>: up
      <1.1.7>: down
      <1.1.8>: up
      <1.1.9>: up
      <1.1.10>: up
      <1.1.11>: up
      <1.1.12>: up
      <1.1.13>: up
      <1.1.14>: up
      ```

   b. **cudbHaState**

Note: If all VMs are fully initialized, then the message "`Status OK`" will be listed in the output of the command under the "SU states" section. If some of the VMs are listed in this section that means the mentioned VM is not fully initialized. In case this happens, wait a few minutes, then execute the command again and see if the status is changed and the VM is not listed anymore.

Contact the next level of maintenance support if one or more VMs fail to initialize in 15 minutes.

7. Start the database cluster processes with the following command:

```
sudo cudbManageStore -a -o start
```

16.3 Node Startup after a Non-Graceful Shutdown

In general, starting a CUDB node after shutting it down requires no special procedures other than powering it on, and handling the alarms that may be raised (and remain raised) after the node is back online. This also applies if the node is restarted after a power failure.

16.3.1 Node Startup after a Non-Graceful Shutdown in Case the CUDB System Is Deployed on Native BSP 8100 Hardware

Compared to a startup after a graceful shutdown, two additional steps must be performed if CUDB nodes deployed on native BSP 8100 hardware are started up after a non-graceful shutdown: due to BSP 8100 characteristics, the blades must be locked before startup, then unlocked when the node is running again.

Refer to the "BSP Equipment Management" document in the BSP 8100 CPI for information on locking blades. See Section 16.2.1 on page 14 for more information on the startup after a graceful shutdown procedure.

Note: If the node has more than one subrack, the subrack with SCs and payload blades must be powered first.

16.3.2 Node Startup after a Non-Graceful Shutdown in Case the CUDB System Is Deployed on a Cloud Infrastructure

In case the CUDB system is deployed on a cloud infrastructure, follow the instructions in Section 16.2.2 on page 16 to start up a node.

16.4 Full CUDB System Startup

In certain cases (such as large-scale power outages) it can happen that the entire CUDB system must be started up.

Note: Full CUDB system startups must be performed by Ericsson personnel only. Contact the next level of maintenance support to perform this procedure.

16.5 Node Graceful Shutdown

Node graceful shutdown allows to manually turn off a CUDB node.

Note: Before performing a graceful shutdown, verify that the CUDB node to turn off does not have any Data Store Unit Group (DSG) or PLDB masters (see Section 18.6 on page 28 for more information). If the node contains a DSG or PLDB master, the mastership must be manually moved to another CUDB node (see Section 18.19 on page 31 for more information).

16.5.1 Node Graceful Shutdown in Case the CUDB System Is Deployed on Native BSP 8100 Hardware

Note: Depending on the number of sites and nodes in the system, shutting down a CUDB node deployed on native BSP 8100 hardware can result in a symmetrical split situation. For example, if a CUDB node is shut down on a 2-node, 2-site deployment, the remaining node finds itself in a symmetrical split situation.

To perform a graceful shutdown, complete the following steps:

1. Establish a CUDB configuration CLI session towards the target CUDB node with the following command:

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

2. Perform a software and configuration backup as described in *CUDB Backup and Restore Procedures*, Reference [9].

3. Stop database cluster processes with the following command:

   ```
   sudo cudbManageStore -a -o stop
   ```

4. Exit the CUDB configuration CLI session.

5. Establish a root CUDB CLI session towards the target CUDB node with the following command:

   ```
   ssh root@<CUDB_Node_OAM_VIP_Address>
   ```

6. Stop the AMF cluster services in the connected SC with the following command:

   ```
   service opensafd stop
   ```

   After the AMF cluster services in the SC are stopped, connect to the other SC (indicated as `SC_2_<x>` below, where `<x>` is the number of the other SC), and stop the AMF cluster services as follows:

```
ssh SC_2_<x>
service opensafd stop
```

Exit the connection to the SC established in this step.

7. Exit the root CUDB CLI session to the SC established in Step 5.
8. Login to the BSP 8100 CLI.
9. Use the CLI console to check which shelf and slot numbers are associated with the SCs and payload blades. Execute the following commands to do so:

```
>configure

(config)>ManagedElement=1,DmxcFunction=1,Eqm=1,Virtua
lEquipment=cudb

(config-VirtualEquipment=cudb)>show-table -m Blade -p
bladeId, userLabel,administrativeState
```

The expected output must be similar to the below example:

```
=============================================
| bladeId | userLabel | administrativeState |
=============================================
| 0-1     | SC-1      | UNLOCKED            |
| 0-11    | PL-6      | UNLOCKED            |
| 0-13    | PL-7      | UNLOCKED            |
| 0-15    | PL-8      | UNLOCKED            |
| 0-17    | PL-9      | UNLOCKED            |
| 0-19    | PL-10     | UNLOCKED            |
| 0-21    | PL-11     | UNLOCKED            |
| 0-23    | PL-12     | UNLOCKED            |
| 0-3     | SC-2      | UNLOCKED            |
| 0-5     | PL-3      | UNLOCKED            |
| 0-7     | PL-4      | UNLOCKED            |
| 0-9     | PL-5      | UNLOCKED            |
=============================================
```

10. Use the CLI console to power off the payload blades and SCs fetched in the previous step. After the payload blades switched off, continue with SCs:

```
(config)>ManagedElement=1,DmxcFunction=1,Eqm=1,Virtua
lEquipment=cudb

(config-VirtualEquipment=cudb)>Blade=0-23,administrat
iveState=LOCKED

(config-VirtualEquipment=cudb)>commit -s

(config-VirtualEquipment=cudb)>Blade=0-21,administrat
iveState=LOCKED

(config-VirtualEquipment=cudb)>commit -s

...
```

```
(config-VirtualEquipment=cudb)>Blade=0-3,administrat
iveState=LOCKED

(config-VirtualEquipment=cudb)>commit -s

(config-VirtualEquipment=cudb)>Blade=0-1,administrat
iveState=LOCKED

(config-VirtualEquipment=cudb)>commit -s

(config-VirtualEquipment=cudb)>end

(VirtualEquipment=cudb)>exit
```

11. Exit the CLI console.

Note: The CUDB CLI session cannot be established once the blades are stopped.

After performing a graceful node shutdown, performing a health check is recommended on the other nodes of the system. For a detailed description on health check, refer to *CUDB Health Check*, Reference [15].

16.5.2 Node Graceful Shutdown in Case the CUDB System Is Deployed on a Cloud Infrastructure

In case of using Cloud Execution Environment (CEE), the graceful shutdown of a CUDB node deployed on a cloud infrastructure can be done through the Atlas GUI or the OpenStack command-line tools. These procedures are described in Section 16.5.2.1 on page 23 and Section 16.5.2.2 on page 24, respectively. For more information about the Atlas GUI, refer to the ‘‘Atlas Dashboard End User Guide’’ document and for more information about the OpenStack command-line tools, refer to the ‘‘OpenStack End User Guide’’ document in the CEE CPI.

In case of using a different cloud solution, refer to the solution-specific documentation for more information.

After performing a graceful node shutdown, performing a health check is recommended on the other nodes of the system. For a detailed description on health check, refer to *CUDB Health Check*, Reference [15].

16.5.2.1 Node Graceful Shutdown Using the Atlas GUI

To gracefully shut down a node through the Atlas GUI, perform the following steps:

1. Stop the database cluster processes with the following command:

   ```
   sudo cudbManageStore -a -o stop
   ```

2. Log in to the Atlas Dashboard.

3. Select the appropriate project in the **Current Project** field and select **Project** in the **View** field.
4. Click the **Instances** category.
5. Mark all payload VMs of the CUDB and choose the **Shut Off Instances** action located in the top bar, under the **More Actions** drop-down menu.
6. Verify that the status of all payload VMs is `Shutoff` and the power state is `Shut Down`.
7. Mark both SCs of the CUDB and choose the **Stop Instances** action located in the top bar, under the **More Actions** drop-down menu.

16.5.2.2 Node Graceful Shutdown Using OpenStack Command-Line Tools

To gracefully shut down a CUDB node through the OpenStack command-line, execute the following commands from CIC:

1. Stop the database cluster processes with the following command:

   ```
   sudo cudbManageStore -a -o stop
   ```

2. Execute the following command to stop payload VMs:

   ```
   nova stop <payload_vm_id>
   ```

   Note: Execute the command for all payload VMs.

3. Execute the following command to stop the SCs of the CUDB:

   ```
   nova stop <SC_id>
   ```

   Note: Execute the command for both SCs.

17 Regular Maintenance Procedures

There are maintenance procedures that need to be performed regularly to ensure the correct operation of CUDB.

These procedures also allow system administrators to detect incidental problems. Refer to *CUDB Node Preventive Maintenance*, Reference [10] for more information on how to detect common problems.

Note: Maintenance of infrastructure equipment is outside the scope of this document.

18 System Level CUDB Procedures

This section describes system-level procedures (such as maintenance operations and scalability tasks) that must be regularly performed in the CUDB system to ensure uninterrupted operation.

Note: Always contact Ericsson support if any problem occurs while performing the following operations.

Warning!

Do not perform operations (such as adding, removing, or including a CUDB node again in the system) that change the number of available CUDB nodes in the system if the `Control, Potential Split Brain Detected` alarm is raised.

Refer to *CUDB High Availability*, Reference [16] for further details on symmetrical split.

18.1 Configuring Network Routes Towards Notification Endpoints

After configuring CUDB notifications, the configuration of the network connectivity facilities is needed in the system for communication towards the endpoints. Refer to *CUDB Notifications*, Reference [18] for more information on how to configure CUDB notifications.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to configure network routes towards notification endpoints.

18.2 Disabling a CUDB Node in a CUDB System

Attention!

Authenticated access to certain CUDB modules may be needed. Contact Ericsson staff support to obtain the access.

This section describes how to disable a CUDB node in a CUDB system.

Note: When disabling a CUDB node that holds 3 BC servers in a two-node site, a split situation will occur in the CUDB system. In this case, it is recommended to manually move all masters from the CUDB site where the CUDB node will be disabled to another CUDB site before disabling the node. For more information on how to manually manage mastership, see Section 18.19 on page 31. In case of a two-site deployment, the system will find itself in a symmetrical split situation.

The procedure of disabling a node in a CUDB system is as follows:

1. Verify that the node to disable (that is, the target node) does not have any DSG or PLDB masters (see Section 18.6 on page 28).

   If the node to disable contains DSG or PLDB masters, the mastership must be manually moved to another CUDB node (see Section 18.19 on page 31).

2. Deactivate the CUDB node locally by modifying the `CudbLocalNode` class in the target node. See Section 19.8 on page 37 for more information.

3. Deactivate the CUDB node remotely by disabling it in the configuration of the rest of CUDB nodes in the system. See Section 20.5 on page 39 for more information.

4. Disable the CUDB Provisioning, CUDB Site, and CUDB Traffic VIP addresses in the local CUDB node. See Section 20.30 on page 47 for more information on this procedure.

18.3 Enabling a Previously Disabled CUDB Node in a CUDB System

If a CUDB node is disabled in a CUDB system, it can be enabled again if the following conditions are met:

- The configuration of the CUDB node to be enabled again (in short, the ‘‘target CUDB node’’) must contain all information related to the rest of the CUDB nodes present in the system.
- The configuration of the rest of the CUDB nodes in the system must contain all the information on the target CUDB node and its PLDB/DSG units.

Perform the following steps to enable a previously disabled CUDB node in a CUDB system:

1. Activate the target node locally. See Section 19.1 on page 34 for more information.

2. Activate the target node in the rest of nodes in the CUDB system. See Section 19.2 on page 34 for more information.

3. Enable the CUDB Provisioning, CUDB SITE, and CUDB Traffic VIP addresses in the local CUDB node. See Section 20.31 on page 48 for more information on this procedure.

18.4 Creating a New DSG

If more storage capacity is needed in a CUDB system, new DSGs must be added.

Note: A drop in LDAP Front End (FE) node and server counters may be observed until the procedure has been fully completed. This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support if new DSG(s) must be created in the CUDB system.

Note: Perform software and configuration backup and system data backup before creating new DSG(s). Follow the procedures to run backups as specified in *CUDB Backup and Restore Procedures*, Reference [9].

18.5 Activating a DSG

If a DSG was created recently, or it was deactivated earlier, it must be activated before using it.

If a CUDB node is added to a CUDB system, the situation can look as follows from the perspective of the DSGs:

- A deactivated DSG is added again to the system.
- A new DSG is added to the system.

Both scenarios are covered in the following sections.

18.5.1 Activating a Previously Deactivated DSG in the CUDB System

The activation procedure for a deactivated DSG in a CUDB system is as follows:

1. Activate the Data Store (DS) Unit which was the DSG master before deactivation. See Section 20.1 on page 38 for more information.
2. Activate the rest of the DS Units in the DSG by following the procedure in Section 20.1 on page 38.

18.5.2 Activating a New DSG in the CUDB System

CUDB supports the activation of new DSGs in the system.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support if new DSG(s) must be activated in the CUDB system.

Warning!

If a new DSG is added to the configuration, always create a new system data backup, because previous backups are not valid any more.

To create a new system data backup, follow the steps described in *CUDB Backup and Restore Procedures*, Reference [9], **Performing System Data Backup** section.

18.6 Listing the Master Replicas

Execute the following command to list all master PLDB and DSG replicas:

```
cudbSystemStatus -R
```

For further information on this command, refer to *CUDB Node Commands and Parameters*, Reference [1].

18.7 Setting the Default Geographical Zone Globally

CUDB supports the global configuration of the default geographical zone.

Perform the procedure described in Section 20.15 on page 42 on every CUDB node to set the default geographical zone. Refer to *CUDB Multiple Geographical Areas*, Reference [17] for more information on geographical zones.

18.8 Setting Custom Distribution Policy for Distribution Entries Globally

CUDB supports the global configuration of custom distribution policies for Distribution Entries (DEs).

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to configure custom distribution policies for DEs globally.

18.9 Checking the Custom Distribution Policy of DEs Globally

CUDB supports the global check of the custom distribution policies configured for DEs.

Perform the procedure described in Section 20.17 on page 42 on every CUDB node to check the custom distribution policy of DEs.

18.10 Restoring the Default Distribution Policy of DEs Globally

CUDB supports the global restore of the custom distribution policies configured for DEs.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to restore custom distribution policies for DEs globally.

18.11 Activating and Deactivating Notifications Globally

CUDB supports the global activation and deactivation of notifications.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to activate or deactivate notifications globally.

18.12 Deactivating Specific Notifications Globally

CUDB supports the global deactivation of specific notifications.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to deactivate specific notifications globally.

18.13 Adding Space to a BLOB in the Disk Storage System Globally

CUDB supports the global increase of space for a Binary Large Object (BLOB) in the disk storage system.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to add space to a BLOB in the disk storage system globally.

18.14 Configuring Subscription Reallocation

Subscription Reallocation (also known as ‘‘reallocation’’) makes it possible to move stored data from one DSG to another.

Reallocation can be executed by moving away a specified percentage of DS entries from source DSG or by providing a list of entries to be moved to the specified destination DSG. Refer to *CUDB Subscription Reallocation*, Reference [19] for more information about reallocation and to *CUDB Node*

Commands and Parameters, Reference [1] for more information about the options of the `cudbReallocate` command.

Note: It is recommended to perform a backup in the CUDB system before running reallocation. Refer to *CUDB Backup and Restore Procedures*, Reference [9] for more information.

It is also recommended to create reconciliation tasks manually after performing reallocation. Create a reconciliation task for each DSG which is a source during the reallocation. Refer to *CUDB Node Commands and Parameters*, Reference [1] for more information about the full syntax of the available reconciliation commands.

Do!

Always execute defragmentation on the source DSG after any kind of reallocation has been applied. See Section 18.16 on page 30 for more information.

18.15 Requesting Reconciliation Manually

Perform the following procedure to request a reconciliation task manually:

1. Find the master DS replica of the affected DSG. See Section 18.6 on page 28 for more information.
2. Perform the procedure described in Section 19.7 on page 37 on the CUDB node hosting the master replica to subscribe the DS master replica to reconciliation.

18.16 Running Defragmentation

Defragmentation is aimed to reorganize the way memory is stored in the PLDB or DS units of the target DSG, to reduce the number of memory gaps.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to perform defragmentation.

18.17 Configuring Provisioning Gateway Nodes

CUDB is able to send notifications to the Ericsson Provisioning Gateway (PG) nodes for provisioning to be stopped during procedures such as system data backup or software upgrade.

For more information on stopping provisioning during backup, refer to *CUDB Backup and Restore Procedures*, Reference [9].

The prerequisite for sending PG notifications is that CUDB nodes are updated with the latest PG configuration information in case there is any change in the PGs connected to the CUDB system. This includes adding or removing PG nodes, or changing the IP address of an existing PG node.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to configure PG nodes.

18.18 CUDB LDAP User Management

If a new CUDB LDAP user or group is added, an existing one is deleted, or the CUDB LDAP user attributes are changed, the LDAP configuration must be updated accordingly. Perform the following procedure to do so:

1. Choose a PLDB Master node where the LDAP user data change is added to the configuration model.
2. Perform one (or more) of the following procedures, depending on the type of LDAP change:
   - To create a new LDAP user, see Section 20.23 on page 44.
   - To delete an existing LDAP user, see Section 20.27 on page 46.
   - To modify an existing LDAP user, see Section 20.24 on page 44.
   - To create a new LDAP user group, see Section 20.22 on page 43.
   - To delete an existing LDAP user group, see Section 20.28 on page 46.
3. The above changes in LDAP user management are automatically propagated to the rest of the CUDB nodes through replication. If that happens, choose another CUDB node and follow the procedure described in Section 20.26 on page 45 .
4. Repeat Step 3 on every CUDB node apart from the one selected in Step 1.

Refer to *CUDB LDAP Interwork Description*, Reference [20] for more information on managing LDAP user attributes.

Note: It is not mandatory to assign CUDB LDAP users to a CUDB LDAP group.

18.19 Changing DSG or PLDB Mastership Manually

This section describes how to move the mastership of a DSG or PLDB to a preferred node as part of a planned master change.

Note: The `cudbSystemStatus` command is used to locate the master replicas.

To perform the mastership change, log in to the target node (where the master replica is to be hosted) and execute the following command:

cudbDsgMastershipChange

Refer to *CUDB Node Commands and Parameters*, Reference [1] for more information on the command.

18.20 Updating LDAP Schema Globally

CUDB supports the global update of the LDAP schema.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to update the LDAP schema globally.

18.21 Activating DS Units

The process of activating a DS Unit depends on whether the unit plays a local or remote role in a CUDB node.

- To activate a local DS Unit in a CUDB node, see Section 20.1 on page 38.
- To activate the same DS Unit in the rest of the CUDB nodes as a remote node, see Section 20.3 on page 39.

18.22 Deactivating DS Units

The process of deactivating a DS Unit depends on whether the unit plays a local or remote role in a CUDB node.

- To deactivate a local DS Unit in a CUDB node, see Section 20.2 on page 38.
- To deactivate the same DS Unit in the rest of the CUDB nodes where it acts as remote node, see Section 20.4 on page 39.

18.23 Configuring Automatic Mastership Change

CUDB is able to automatically change the master to the highest priority DSG or PLDB replica, when the master replica is in a different DSG or PLDB replica. For more information, refer to *CUDB High Availability*, Reference [16].

To configure Automatic Mastership Change (AMC), modify the `CudbAutomaticMasterChange` class (refer to the ‘‘Class CudbAutomaticMasterChange’’ section of *CUDB Node Configuration Data Model Description*, Reference [3]) in the configuration model as follows:

- To enable or disable AMC, set the `enabled` attribute to the corresponding value of `true` or `false`.

- To change the maximum replication delay, modify the `maxReplicationTimeDelay` attribute. The value of the attribute is the number of milliseconds. Refer to the ‘‘Class CudbAutomaticMasterChange’’ section of *CUDB Node Configuration Data Model Description*, Reference [3] for the default value.

- To change the time window, in which AMC is running, modify the attributes `timeWindowStart` and `timeWindowEnd`, as described in the ‘‘Class CudbAutomaticMasterChange’’ section of *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 administrative operation `applyConfig` to activate the changes).

18.24 LDAP Data Views Management

The LDAP Data Views function supports accessing stored data through customizable views. The function is based on assigning views to LDAP users. If the LDAP user sending a request has an LDAP view assigned by configuration, then that user can access the data through that view.

Refer to *CUDB LDAP Data Views Management*, Reference [21] for detailed information on all the steps required to create, configure, and delete an LDAP view.

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

18.25 OAM Centralized Authentication System Support

CUDB allows to define and manage new OAM users and groups in remote LDAP servers only, in a centralized way, instead of having to create them locally on all CUDB nodes.

Refer to *CUDB Security and Privacy Management*, Reference [8] for more information on how to create and authenticate remote OAM users.

19 Node Level CUDB Procedures

This section describes node-level procedures that may need to be performed in the CUDB system.

Note: Always contact Ericsson support if any problem occurs while performing the following operations.

19.1 Activating a Local CUDB Node

When a CUDB node is installed, it must be activated before using it.

To activate a CUDB node locally, set the value of the `enabled` attribute of the corresponding instance of the `CudbLocalNode` class to `true`. For more information, refer to the "Class CudbLocalNode" section of *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 administrative operation `applyConfig` to activate the changes).

19.2 Activating a Remote CUDB Node

Before activating a previously deactivated CUDB node locally, it must be activated remotely in the rest of the CUDB nodes of the system.

To activate a CUDB node remotely, set the value of the `enabled` attribute of the corresponding instance of the `CudbRemoteNode` class to `true`. For more information, refer to the "Class CudbRemoteNode" section of *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 administrative operation `applyConfig` to activate the changes).

19.3 Modifying a CUDB Node Name

The name of a CUDB node can be modified after installation.

To modify the name of a CUDB node, change the value of the `networkElementName` attribute of the corresponding instance of the `CudbLocalNode` class. For more information, refer to the "Class CudbLocalNode" section of *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 administrative operation `applyConfig` to activate the changes).

19.4 Modifying the DSG or PLDB Cluster State

The procedure to modify the state of a DSG or PLDB cluster is the following:

1. Establish a CUDB CLI session towards the target CUDB node with the following command:

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

2. Execute the `cudbManageStore` command on the target cluster to modify its state to the new state. In case of a DSG cluster, use the command as follows:

   ```
   sudo cudbManageStore --ds <dsId> --order <New_state>
   ```

   In case of a PLDB cluster, use the command as follows:

   ```
   sudo cudbManageStore --pl --order <New_state>
   ```

   Refer to *CUDB Node Commands and Parameters*, Reference [1] for detailed information on cluster states.

3. Exit the CUDB CLI session with the following command:

   ```
   exit
   ```

Note: Changing the state of a cluster in a way that renders the cluster offline (such as changing to maintenance mode, or restoring an earlier backup) can have serious consequences.

If a DSG cluster containing a master replica goes offline, a new DSG master is selected. However, if the PLDB cluster goes offline, the corresponding CUDB node also goes offline.

19.5 Checking Software Inventory on a CUDB Node

CUDB provides several ways to check what software is installed on a CUDB node. These methods are described in the next sections.

19.5.1 Checking Imported Software on a CUDB Node

CUDB provides a mechanism to check which packages have been imported to a CUDB node. To check the Software Inventory of a CUDB node, perform the following steps:

1. Establish a CUDB CLI session towards the target CUDB node with the following command:

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

2. Check the software packages installed in the CUDB node with the following command:

   cmw-repository-list

   Note: The above procedure lists packages either as “Used” or “Not used”. Packages listed as “Used” are installed packages, while packages listed as “Not used” are packages that have been imported, but not yet installed.

3. Exit the CUDB CLI session with the following command:

   exit

19.5.2 Checking Installed Software on Each Blade or VM of a CUDB Node

CUDB provides a mechanism to check what packages are installed on each blade or VM of a CUDB node. To check the Software Inventory in a specific blade or VM of a CUDB node, perform the following steps:

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

   ssh *<admin_user>@<CUDB_Node_OAM_VIP_Address>*

2. Check the software packages installed in the CUDB node with the following command:

   sudo cmw-rpm-list *<CUDB_node_blade_or_vm_host_name>*

   In the above command, the ***<CUDB_node_blade_or_vm_host_name>*** variable is used to indicate the blade or VM host name to be checked. If left empty, all blades or VMs are listed. The host name can be the following:

   - `SC_2_1` : First SC.
   - `SC_2_2` : Second SC.
   - `PL_2_<n>` : Payload blades or VMs. The variable *<n>* stands for the payload blade or VM number that must be set from the highest to the lowest value, according to the configured blades or VMs. *<n>* can have a value of up to 40.

3. Exit the CUDB CLI session with the following command:

   exit

19.6 Defragmenting a Database Cluster

The CUDB system supports the defragmentation of individual database clusters.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to defragment CUDB nodes.

19.7 Subscribing a DSG Master Replica to Reconciliation Manually

Perform the following steps to subscribe a DSG master replica to reconciliation manually:

1. Establish a new administrative CUDB CLI session towards the target CUDB node with the following command:

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

2. Execute the following command:

   ```
   sudo cudbReconciliationMgr --add <dsg_id>
   ```

   In the above command, `<dsg_id>` is the identifier of the DSG to which the target master DSG replica belongs.

   Refer to *CUDB Node Commands and Parameters*, Reference [1] for further information on this command.

19.8 Deactivating a Local CUDB Node

To deactivate a CUDB node locally, set the value of the `enabled` attribute of the corresponding instance of the `CudbLocalNode` class to `false`. For more information, refer to the ‘‘Class CudbLocalNode’’ section of *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 administrative operation `applyConfig` to activate the changes).

19.9 Storage Performance Monitoring Function

The CUDB system can detect storage system failures on SCs and payload blades or VMs by two mechanisms:

- Monitoring I/O heavy processes (the time spent in D state).
- Probing the file system.

CUDB supports the activation, deactivation of the Storage Performance Monitoring function, and the configuration of the function parameters by modifying the `/home/cudb/monitoring/blade/config/cudbHwFaultReaction.json` configuration file. Storage Performance Monitoring is enabled by default.

Note: This configuration file can be changed by Ericsson personnel only.

20 Auxiliary CUDB Procedures

The procedures described in this section are required to properly maintain a CUDB system, but are not supposed to be executed in a standalone manner. Instead, they are parts of higher-level, more complex procedures, such as those described in Section 18 on page 24 and Section 19 on page 33.

20.1 Activating a DS Unit in a Local CUDB Node

Before using a DS Unit in a DSG, it must be activated.

To activate a DS Unit in a local CUDB node, set the value of the `enabled` attribute of the corresponding instance of the `CudbLocalDs` class to `true`. For more information, refer to the ‘‘Class CudbLocalDs’’ section of *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 administrative operation `applyConfig` to activate the changes).

20.2 Deactivating a DS Unit in a Local CUDB Node

DS Units in a CUDB node are usually deactivated for maintenance reasons.

To deactivate a DS Unit in a local CUDB node, set the value of the `enabled` attribute of the corresponding instance of the `CudbLocalDs` class to `false`. For more information, refer to the ‘‘Class CudbLocalDs’’ section of *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 administrative operation `applyConfig` to activate the changes).

Warning!

Even if a DS Unit is deactivated, it continues to operate. Therefore, it can raise alarms even if being deactivated.

20.3 Activating a DS Unit in a Remote CUDB Node

To activate a DS Unit in a remote CUDB node, set the value of the `enabled` attribute of the corresponding instance of the `CudbRemoteDs` class to `true`. For more information, refer to the ''Class CudbRemoteDs'' section of *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 administrative operation `applyConfig` to activate the changes).

20.4 Deactivating a DS Unit in a Remote CUDB Node

To deactivate a remote DS Unit, set the value of the `enabled` attribute of the corresponding instance of the `CudbRemoteDs` class to `false`. For more information, refer to the ''Class CudbRemoteDs'' section of *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 administrative operation `applyConfig` to activate the changes).

Warning!

Even if a DS Unit is deactivated, it continues to operate. Therefore, it can raise alarms even if being deactivated.

20.5 Deactivating a Remote CUDB Node

After a CUDB node is deactivated locally, it is necessary to deactivate it also in the rest of CUDB nodes.

To deactivate a remote CUDB node, set the value of the `enabled` attribute of the corresponding instance of the `CudbRemoteNode` class to `false`. For more

information, refer to the ‘‘Class CudbRemoteNode’’ section of *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 administrative operation `applyConfig` to activate the changes).

20.6 Adding a DSG to a Local CUDB Node

CUDB supports adding new DSGs to a local CUDB node.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to add new DSGs to a local CUDB node.

20.7 Adding a DS Unit to a Local CUDB Node

When a new DSG is created or the geographical redundancy of the system is upgraded, the new DS Units must be added to the CUDB node configuration where the new DS Unit is hosted.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to add a DS Unit to a local CUDB node.

20.8 Adding a Generic Blade or VM to a Local CUDB Node

CUDB supports adding new blades or VMs to local CUDB nodes.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to add new blades or VMs to CUDB nodes.

20.9 Configuring a DS Unit Added to a Local CUDB Node

After new blades or VMs were added to a CUDB node, the DS Unit must be configured.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to configure the new blades or VMs added to the CUDB node.

20.10 Configuring a DS Unit Added to a Remote CUDB Node

When a new DS Unit is added to the CUDB system, it must be configured both on its local node, and also on the rest of the CUDB nodes to recognize it in the entire system.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to configure the new blades or VMs added to the CUDB node.

- `cudbRemoteDsId`
- `dsGroupId`
- `enabled`

For more information on the `CudbRemoteDs` class, refer to the ‘‘Class CudbRemoteDs’’ section of *CUDB Node Configuration Data Model Description*, Reference [3].

20.11 Deactivating a Local PLDB Unit

To deactivate a local PLDB unit in a CUDB node, set the value of the `enabled` attribute of the corresponding instance of the `CudbLocalPl` class to `false`. For more information, refer to the ‘‘Class CudbLocalPl’’ section of *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 administrative operation `applyConfig` to activate the changes).

20.12 Activating a Local PLDB Unit

To activate a PLDB unit in a CUDB node, set the value of the `enabled` attribute of the corresponding instance of the `CudbLocalPl` class to `true`. For more information, refer to the ‘‘Class CudbLocalPl’’ section of *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 administrative operation `applyConfig` to activate the changes).

20.13 Deactivating a Remote PLDB Unit

To deactivate a PLDB unit belonging to another CUDB node, set the value of the `enabled` attribute of the corresponding instance of the `CudbRemotePl` class to `false`. For more information, refer to the ‘‘Class CudbRemotePl’’ section of *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 administrative operation `applyConfig` to activate the changes).

20.14 Activating a Remote PLDB Unit

To activate a PLDB unit belonging to another CUDB node, set the value of the `enabled` attribute of the corresponding instance of the `CudbRemotePl` class to `true`. For more information, refer to the "Class CudbRemotePl" section of *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 administrative operation `applyConfig` to activate the changes).

20.15 Setting the Default Geographical Zone

To set the default geographical zone in a specific CUDB node, set the value of the `defaultZone` attribute of the corresponding instance of the `CudbSystem` class to the new default geographical zone. For more information, refer to the "Class CudbSystem" section of *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 administrative operation `applyConfig` to activate the changes).

Note: Setting the default zone is only possible if the Deployment Flexibility Value Package is available.

20.16 Setting Custom Distribution Policy for DEs

CUDB supports the configuration of custom distribution policies for DEs in the CUDB nodes.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to set custom distribution policy for DEs.

20.17 Checking The Custom Distribution Policy of DEs

To check whether a specific CUDB node is using the default distribution policy or a custom one, perform the following steps:

1. Establish a CUDB CLI session with the following command towards the CUDB node where the distribution policy is to be checked:

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

2. Check the status of the distribution policy with the following command:

   ```
   sudo cudbManageLibDataDist --status
   ```

   The output of the command clearly states if a custom distribution policy is being applied.

3. Exit the CUDB CLI with the following command:

   ```
   exit
   ```

20.18 Restoring the Distribution Policy of DEs

CUDB supports restoring the default distribution policy of DEs.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to restore the distribution policy of DEs.

20.19 Increasing the Space of a BLOB in the Disk Storage System

CUDB supports increasing the space of a BLOB in the disk storage system.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to increase the space of a BLOB in the disk storage system.

20.20 Enabling or Disabling Notifications for All Applications

CUDB supports enabling or disabling notifications for all applications.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to enable or disable notifications.

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 administrative operation `applyConfig` to activate the changes).

20.21 Disabling Notifications towards a Specification Application End Point

CUDB can prevent the sending of notifications to a particular application Front End (FE) even if notifications are globally enabled.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to disable notifications for a specific application.

20.22 Creating a CUDB LDAP User Group in a Local CUDB Node

To create a new CUDB LDAP user in a local CUDB node, add a new instance of the `CudbLdapUserGroup` class to the configuration model. For more information, refer to the ‘‘Class CudbLdapUserGroup’’ section of *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 administrative operation `applyConfig` to activate the changes).

20.23 Adding a New CUDB LDAP User to a Local CUDB Node

To add a new CUDB LDAP user to a local CUDB node, add a new instance of the `CudbLdapUser` class to the configuration model and set the following mandatory attributes:

- `cudbUserPassword`
- `cudbUserGroup`
- `readModeInPL`
- `readModeInDS`

Additional optional attributes can also be configured for the LDAP user, each in a new line.

For more information on the `CudbLdapUser` class, refer to the ‘‘Class CudbLdapUser’’ section of *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 administrative operation `applyConfig` to activate the changes).

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

- `readModeInPL=LP` (Local Preferred) and `readModeInDS=MP` (Master Preferred).
- `readModeInPL=MA` and `readModeInDS=MA` (both Master Always).

20.24 Configuring Attributes of a CUDB LDAP User

Once a CUDB LDAP user is created, a set of attributes can be modified. These attributes are listed below:

```
'countersGroup'

'readModeInPL'

'readModeInDS'

'isProvisioningUser'

'isReProvisioningUser'

'userLdapAuth'

'userLdapHash'

'overloadRejectionWeight'

'cudbUserPassword'
```

Changes for the `userLdapAuth`, `userLdapHash`, and `cudbUserPassword` attributes are described in *CUDB Security and Privacy Management*, Reference [8].

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 administrative operation `applyConfig` to activate the changes).

20.25 Configuring a PG Node in a Local CUDB Node for Backup Notification

PG nodes are configured in CUDB nodes as the destination for backup notifications.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to add new CUDB LDAP users to CUDB nodes.

Nodes are separated with “,” and addresses for each node are separated with “;”.

20.26 Updating CUDB LDAP User Information in a CUDB Node

To update CUDB LDAP user information in a remote CUDB node, perform the following steps:

1. Execute the administrative operation `updateUserInfo`. Refer to the "updateUserInfo" section of *CUDB Node Configuration Data Model Description*, Reference [3] for more information.

2. After executing `updateUserInfo`, check the `CudbLdapUser` class in the configuration model to see whether the change was applied correctly.

   For more information on how to perform this check, refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [3].

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

3. If the changes were not applied, then the CUDB node has not been informed of the update yet. In this case, execute `updateUserInfo` again.

20.27 Deleting a CUDB LDAP User in a Local CUDB Node

To delete a CUDB LDAP user from a local CUDB node, delete the appropriate instance of the `CudbLdapUser` class from the configuration model. For more information, refer to the "Class CudbLdapUser" section of *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 administrative operation `applyConfig` to activate the changes).

20.28 Deleting a CUDB LDAP User Group in a Local CUDB Node

To delete a CUDB LDAP user in a local CUDB node, delete the appropriate instance of the `CudbLdapUserGroup` class from the configuration model. For more information, refer to the "Class CudbLdapUserGroup" section of *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 administrative operation `applyConfig` to activate the changes).

20.29 Removing Huge Files

The removal of huge files with the `rm` Linux command results in instability in blades or VMs. Also, if executed on the SC, input/output resources cannot be accessed by the processes running on it.

Therefore, to avoid stability and performance issues, it is strongly recommended to use the following command instead of `rm` to remove huge files:

```
ionice -c 3 ls -1 <files to remove> | sed 's/\(.*\)/sleep 1
\&\& ionice -c 3 rm -fv \1/g' | bash
```

20.30 Disabling Provisioning, Traffic, and Site VIP Addresses

To disable Provisioning, Traffic, and Site VIP addresses, perform the following steps:

1. Set the value of the `adminState` attribute of the *`CudbTrafficControlManager`* class to UNLOCKED.

   Note: Make sure that this step is executed before moving on to the next steps.

2. Create a new blocking rule for the Provisioning VIP address:

   a. Add a new instance of the *`CudbTrafficBlockingRule`* class under the *`CudbTrafficControlManager`* class.

   b. Set the value of the `blockedVIP` attribute to the Provisioning VIP address that can be obtained from the network plan.

3. Create a new blocking rule for the Traffic VIP address:

   a. Add a new instance of the *`CudbTrafficBlockingRule`* class under the *`CudbTrafficControlManager`* class.

   b. Set the value of the `blockedVIP` attribute to the Traffic VIP address that can be obtained from the `trafficVIP` attribute of the *`CudbLocalNode`* class.

4. Create a new blocking rule for the Site VIP address:

   a. Add a new instance of the *`CudbTrafficBlockingRule`* class under the *`CudbTrafficControlManager`* class.

   b. Set the value of the `blockedVIP` attribute to the Site VIP address that can be obtained from the `cudbVIP` attribute of the *`CudbLocalNode`* class.

5. Check the `trafficControlManagerState` attribute of the `CudbTrafficControlManager` class.

If `trafficControlManagerState` is `DISABLED`, then a problem occurred during the activation of the configuration change in the node, and the node behavior cannot be guaranteed to be consistent with the configuration.

Note: The `cudbTrafficBlockingRuleId` attribute must be unique for each newly created blocking rule.

For more information on the `CudbTrafficBlockingRule` class, refer to the "Class CudbTrafficBlockingRule" section of *CUDB Node Configuration Data Model Description*, Reference [3].

For more information on the `CudbTrafficControlManager` class, refer to the "Class CudbTrafficControlManager" section of *CUDB Node Configuration Data Model Description*, Reference [3].

For more information on the `CudbLocalNode` class, refer to the "Class CudbLocalNode" section of *CUDB Node Configuration Data Model Description*, Reference [3].

Example 1 (shown in COM CLI) shows the commands for disabling Provisioning, Traffic, and Site VIP addresses of node 100 with 10.10.10.1 configured as Provisioning VIP, 10.10.10.10 configured as Traffic VIP, and 10.10.10.20 configured as Site VIP.

```
>configure
(config)>ManagedElement=1,CudbSystem=1,CudbLocalNode=100,CudbTrafficControlManager=1
(config-CudbTrafficControlManager=1)>adminState=UNLOCKED
(config-CudbTrafficControlManager=1)>CudbTrafficBlockingRule=1
(config-CudbTrafficBlockingRule=1)>blockedVIP="10.10.10.1"
(config-CudbTrafficBlockingRule=1)>up
(config-CudbTrafficControlManager=1)>CudbTrafficBlockingRule=2
(config-CudbTrafficBlockingRule=2)>blockedVIP="10.10.10.10"
(config-CudbTrafficBlockingRule=2)>up
(config-CudbTrafficControlManager=1)>CudbTrafficBlockingRule=3
(config-CudbTrafficBlockingRule=3)>blockedVIP="10.10.10.20"
(config-CudbTrafficBlockingRule=3)>commit
(CudbTrafficBlockingRule=3)>up
(CudbTrafficControlManager=1)>show all
CudbTrafficControlManager=1
adminState=UNLOCKED
trafficControlManagerState=ENABLED
CudbTrafficBlockingRule=1
blockedVIP="10.10.10.1"
CudbTrafficBlockingRule=2
blockedVIP="10.10.10.10"
CudbTrafficBlockingRule=3
blockedVIP="10.10.10.20"
(CudbTrafficControlManager=1)>exit
```

Example 1 Disabling Addresses

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 administrative operation `applyConfig` to activate the changes).

20.31 Enabling Provisioning, Traffic, and Site VIP Addresses

To re-enable Provisioning, Traffic, and Site VIP addresses, perform the following steps:

1. To list all present blocking rules, show all instances of the *CudbTrafficBlockingRule* class under the *CudbTrafficControlManager* class, together with their attributes.

2. Remove the blocking rule for the Provisioning VIP address:

    a. Identify which blocking rule is for the Provisioning VIP address. To do that, examine the output of Step 1 and look for the instance of the *CudbTrafficBlockingRule* class where the value of the `blockedVIP` attribute is equal to the Provisioning VIP address that can be obtained from the network plan.

    b. Remove the identified instance of the *CudbTrafficBlockingRule* class.

3. Remove the blocking rule for the Traffic VIP address:

    a. Identify which blocking rule is for the Traffic VIP address. To do that, examine the output of Step 1 and look for the instance of the *CudbTrafficBlockingRule* class where the value of the `blockedVIP` attribute is equal to the Traffic VIP address that can be obtained from the `trafficVIP` attribute of the *CudbLocalNode* class.

    b. Remove the identified instance of the *CudbTrafficBlockingRule* class.

4. Remove the blocking rule for the Site VIP address:

    a. Identify which blocking rule is for the Site VIP address. To do that, examine the output of Step 1 and look for the instance of the *CudbTrafficBlockingRule* class where the value of the `blockedVIP` attribute is equal to the Site VIP address that can be obtained from the `cudbVIP` attribute of the *CudbLocalNode* class.

    b. Remove the identified instance of the *CudbTrafficBlockingRule* class.

5. Set the value of the `adminState` attribute of the to *CudbTrafficControlManager* class to `LOCKED`.

    Note: Make sure that Step 2, Step 3, and Step 4 are executed previously, because `adminState` cannot be set to `LOCKED` if there is any blocking rule present.

6. Check the `trafficControlManagerState` attribute of the `CudbTrafficControlManager` class.

If `trafficControlManagerState` is `DISABLED`, then a problem occurred during the activation of the configuration change in the node, and the node behavior cannot be guaranteed to be consistent with the configuration.

For more information on the `CudbTrafficBlockingRule` class, refer to the "Class CudbTrafficBlockingRule" section of *CUDB Node Configuration Data Model Description*, Reference [3].

For more information on the `CudbTrafficControlManager` class, refer to the "Class CudbTrafficControlManager" section of *CUDB Node Configuration Data Model Description*, Reference [3].

For more information on the `CudbLocalNode` class, refer to the "Class CudbLocalNode" section of *CUDB Node Configuration Data Model Description*, Reference [3].

Example 2 (shown in COM CLI) shows the commands for re-enabling all VIP addresses of node 100.

```
>configure

(config)>ManagedElement=1,CudbSystem=1,CudbLocalNode=100,CudbTrafficControlManager=1
(config-CudbTrafficControlManager=1)>show all verbose
CudbTrafficControlManager=1
adminState=UNLOCKED
cudbTrafficControlManagerId="1"
trafficControlManagerState=ENABLED <read-only>
CudbTrafficBlockingRule=1
blockedVIP="10.10.10.1"
cudbTrafficBlockingRuleId="1"
CudbTrafficBlockingRule=2
blockedVIP="10.10.10.10"
cudbTrafficBlockingRuleId="2"
CudbTrafficBlockingRule=3
blockedVIP="10.10.10.20"
cudbTrafficBlockingRuleId="3"



(config-CudbTrafficControlManager=1)>adminState=UNLOCKED
(config-CudbTrafficControlManager=1)>no CudbTrafficBlockingRule=1
(config-CudbTrafficControlManager=1)>no CudbTrafficBlockingRule=2
(config-CudbTrafficControlManager=1)>no CudbTrafficBlockingRule=3
(config-CudbTrafficControlManager=1)>adminState=LOCKED
(config-CudbTrafficControlManager=1)>commit
(CudbTrafficControlManager=1)>show all
CudbTrafficControlManager=1
adminState=LOCKED
trafficControlManagerState=ENABLED
(CudbTrafficControlManager=1)>exit
```

Example 2 Re-Enabling VIP Addresses

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 administrative operation `applyConfig` to activate the changes).

20.32 Configuring PG Endpoints in a Local CUDB Node for the Provisioning Assurance Function

PG endpoints are configured in the CUDB nodes as destinations for the Provisioning Assurance function interworking.

Note: This procedure can be performed by Ericsson personnel only. Contact the next level of maintenance support to add new CUDB LDAP users to CUDB nodes.

20.33 Creating a New Notification Event

To create a new CUDB Notification Event, a new instance of the `CudbNotificationEvent` class must be added to the configuration model and the corresponding attributes must be set. For more information, refer to the ‘‘Class CudbNotificationEvent’’ section of *CUDB Node Configuration Data Model Description*, Reference [3].

Below is an example, shown in COM CLI, of `CudBNotificationEvent=1` with `eventId=”SAE-HLR”` with defined one Notification Endpoint and five Notification Object Classes:

```
>show verbose ManagedElement=1,CudbSystem=1,CudbNotifications=1,CudbNotificationEvent=1
CudbNotificationEvent=1
   cudbNotificationEventId="1"
   eventId="SAE-HLR"
   notificationString="mobilityEvent"
   userLabel=[] <empty>
   CudbNotificationEndPoint=1
   CudbNotificationObjectClass=1
   CudbNotificationObjectClass=2
   CudbNotificationObjectClass=3
   CudbNotificationObjectClass=4
   CudbNotificationObjectClass=5
```

Define all required elements (attributes, object/object classes and their attributes) of the new `CudbNotificationEvent` in its tree structure (Directory Information Tree, DIT) before starting to create a new Notification Event.

Following is an example of a new `CudbNotificationEvent=2` shown in COM CLI:

```
>show verbose ManagedElement=1,CudbSystem=1,CudbNotifications=1,CudbNotificationEvent=2
CudbNotificationEvent=2
   cudbNotificationEventId="2"
   eventId="SAE-HSS"
   notificationString="mobilityEvent"
   userLabel=[] <empty>
   CudbNotificationEndPoint=1
   CudbNotificationObjectClass=1

>show verbose ManagedElement=1,CudbSystem=1,CudbNotifications=1,CudbNotificationEvent=2,CudbNotificationEndPoint=
CudbNotificationEndPoint=1
   name="Serv1"
   URI="http://127.0.0.1:8080"
   webService="/"
   weight=1

>show verbose ManagedElement=1,CudbSystem=1,CudbNotifications=1,CudbNotificationEvent=2,CudbNotificationObjectCla
CudbNotificationObjectClass=1
   cudbNotificationObjectClassId="1"
   dn="serv=csps"
   name="CsPsLocationData"
   type="related"
   userLabel=[] <empty>
   CudbNotificationAttr=1

>show verbose ManagedElement=1,CudbSystem=1,CudbNotifications=1,CudbNotificationEvent=2,⇒
CudbNotificationObjectClass=1,CudbNotificationAttr=1
CudbNotificationAttr=1
   cudbNotificationAttrId="1"
   name="SGSNNUM"
   send=true
   userLabel=[] <empty>
   value=""
(CudbNotificationObjectClass=1)>
```

As shown above, `CudbNotificationEvent=2` will contain one `CudbNotificationEndpoint=1` and one `CudbNotificationObjectClass=1` with one `CudbNotificationAttr=1`.

Follow the steps below, shown in COM CLI, to insert a new Notification Event using the Object Model Modification procedure:

1. Establish a CUDB CLI session towards the CUDB node by executing the `ssh -l cudbadmin` *<CUDB_Node_OAM_IP_Address>* command.

   Important: This procedure must be done in all CUDB nodes in the system.

2. If there is no backup of the present configuration, perform the backup by executing the `sudo cmw-configuration-persist` command. If the backup is already updated, proceed to Step 3.

3. Establish a CUDB configuration CLI session in the active SC. Use the commands below (their output is also shown) to find the active SC:
   `sudo cudbHaState | grep COM | grep ACTIVE`
   `COM is assigned as ACTIVE in controller SC-1`

   The active SC, (SC_2_1 in the example above) must be used for accessing the COM CLI by executing the `sudo /opt/com/bin/cliss` command.

4. Set the configuration session by executing the `configure` command.

5. Check if `CudbNotifications` are already enabled and exist in `CudbNotificationsEvents`.

Use the `show verbose` command to show `ManagedElement=1,CudbSystem=1,CudbNotifications=1`:

```
(config)>show verbose ManagedElement=1,CudbSystem=1,CudbNotifications=1
CudbNotifications=1
cudbNotificationsId="1"
enabled=true <default>
maxReattempts=3 <default>
reattemptTime=1000 <default>
userLabel=[] <empty>
CudbNotificationEvent=1
(config)>
```

If `CudbNotifications` are not already enabled, execute the following commands:

```
(config)>ManagedElement=1,CudbSystem=1,CudbNotifications=1,enabled=true
(config)>commit
```

6. Execute the following command sequence to add a new CUDB Notification Event (CudbNotificationEvent=2) in the CUDB configuration model:

   Note: This is an example and must not be taken as a rule.

```
>configure
(config)>ManagedElement=1,CudbSystem=1,CudbNotifications=1,CudbNotificationEvent=2
(config-CudbNotificationEvent=2)>eventId=SAE-HSS
(config-CudbNotificationEvent=2)>notificationString=mobilityEvent
(config-CudbNotificationEvent=2)>CudbNotificationEndPoint=1
(config-CudbNotificationEndPoint=1)>name=Serv1
(config-CudbNotificationEndPoint=1)>URI=http://127.0.0.1:8080
(config-CudbNotificationEndPoint=1)>weight=1
(config-CudbNotificationEndPoint=1)>up
(config-CudbNotificationEvent=2)>CudbNotificationObjectClass=1
(config-CudbNotificationObjectClass=1)>dn="serv=csps"
(config-CudbNotificationObjectClass=1)>name=CsPsLocationData
(config-CudbNotificationObjectClass=1)>type=related
(config-CudbNotificationObjectClass=1)>CudbNotificationAttr=1
(config-CudbNotificationAttr=1)>name=SGSNNUM
(config-CudbNotificationAttr=1)>send=true
(config-CudbNotificationAttr=1)>up
(config-CudbNotificationObjectClass=1)>up
(config-CudbNotificationEvent=2)>show verbose CudbNotificationEvent=2
```

```
cudbNotificationEventId="2"
eventId="SAE-HSS"
notificationString="mobilityEvent"
userLabel=[] <empty>
CudbNotificationEndPoint=1
CudbNotificationObjectClass=1
```

Each operation is executed as a unique transaction.

7. Commit the changes by executing the `commit` command.
8. Check the log files to see the result of the operations. For more information, see *CUDB Node Logging Events*, Reference [7].
9. Check the configuration changes with the `show ManagedElement=1,CudbSystem=1,CudbNotificationEvent=2` command.

   Note: Remember to use `show verbose` instead of `show` for not mandatory attributes that must have no set value, or for optional attributes whose value is set to the default one.

10. Exit from the configuration mode by executing the `end` command.
11. Exit from the CUDB configuration CLI console by executing the `exit` command.

Note: Configuration changes in the examples above can also be performed through the NETCONF client.

Refer to *CUDB Notifications*, Reference [18] for more information about Notifications Parameters.

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 administrative operation `applyConfig` to activate the changes).

20.34 Configuring QoS in CUDB

In CUDB the Differentiated Services (DiffServ) computer networking architecture is used to provide QoS (Quality of Service). The basic principle of DiffServ is the classification of IP packets by applying a special marking on them, that is placing a DSCP mark in the IPv4 TOS field. Based on this mark, the DiffServ-Aware routers in the DiffServ domain can handle the traffic (Per Hop Behavior) depending on the traffic class which the packet belongs to. After that, QoS is provided by `iptables`. Using this method, the outgoing packages are marked with the preconfigured DSCP values. Refer to *CUDB Node Network Description*, Reference [12] for more information on the preconfigured values.

Note: Configuring QoS is restricted to Ericsson personnel. Contact the next level of maintenance support to perform such procedures.

Glossary

For the terms, definitions, acronyms and abbreviations used in this document, refer to *CUDB Glossary of Terms and Acronyms*, Reference [22].

Reference List

CUDB Documents

[1] *CUDB Node Commands and Parameters*

[2] *CUDB Users and Passwords*, 3/00651-HDA 104 03/9

[3] *CUDB Node Configuration Data Model Description*

[4] *CUDB Node Fault Management Configuration Guide*

[5] *CUDB Counters List*

[6] *CUDB Performance Guide*

[7] *CUDB Node Logging Events*

[8] *CUDB Security and Privacy Management*

[9] *CUDB Backup and Restore Procedures*

[10] *CUDB Node Preventive Maintenance*

[11] *CUDB Data Storage Handling*

[12] *CUDB Node Network Description*

[13] *CUDB Import and Export Procedures*

[14] *CUDB LDAP Schema Management Graphical User Interface*

[15] *CUDB Health Check*

[16] *CUDB High Availability*

[17] *CUDB Multiple Geographical Areas*

[18] *CUDB Notifications*

[19] *CUDB Subscription Reallocation*

[20] *CUDB LDAP Interwork Description*

[21] *CUDB LDAP Data Views Management*

[22] *CUDB Glossary of Terms and Acronyms*

Other Ericsson Documents

[23] *COM Management Guide*

[24] *COM NETCONF Interface Base*

[25] *LDE Management Guide*

[26] *eVIP Management Guide*

Other Documents and Online References

[27] *OpenLDAP, ldapsearch Tool Manual*