ERICSSON

CUDB Troubleshooting Guide

CUDB TROUBLESHOOTING GUIDE

1/1553-HDA 104 03/10-V1 Uen A

Contents

1 Introduction

This document provides troubleshooting information for CUDB nodes. However, since a CUDB system is made up of a set of CUDB nodes, the document is also valid for troubleshooting system-level problems and failures.

1.1 Scope

The purpose of this document is to provide the instructions and tools needed to recover a CUDB node in case of abnormal behavior or failures.

The document does not contain information on the maintenance tasks and configuration procedures, refer to *CUDB Node Preventive Maintenance*, Reference [1] for more information on these topics.

For more information on the user names and passwords used in this document, refer to ‘‘CUDB Users and Passwords’’, Reference [2].

This document is intended for personnel working with the CUDB system.

Note: In case the procedures described in this document do not fix the experienced faults, contact the next level of Ericsson support.

1.2 Revision Information

Rev. A

This document is based on 1/1553-HDA 104 03/9 with the following changes:

- Updated hardware information.
- Virtualization terminology update all throughout the document.
- Section 2.1.6 on page 6: Expanded the masterlist description with information on split brain situations. Added information on replication command output. Updated output description.

1.3 Prerequisites

This document provides troubleshooting information only for properly installed and configured CUDB nodes. Therefore, it is assumed that the CUDB node was installed and configured appropriately before the experienced abnormal behavior or failure occurred.

1.3.1 Required Documents

Before starting the troubleshooting procedure, ensure that the following information or documents are available:

- This document.
- All the documents listed in the References section. (See Reference List on page 19).

1.3.2 Conditions

Certain troubleshooting activities can have an impact on node performance. For example, trace or log activation can affect traffic throughput and is not recommended without first consulting Ericsson.

1.4 Related Information

Definition and explanation of acronyms and terminology, trademark information, and typographic conventions can be found in the following documents:

- *CUDB Glossary of Terms and Acronyms*, Reference [3]
- *Trademark Information*, Reference [4]
- *Typographic Conventions*, Reference [5]

2 Troubleshooting Tools

This section describes the tools that can be used to troubleshoot the CUDB system.

2.1 CLI Commands

This section describes the tools and resources that can be used to help troubleshoot CUDB through the command line interface of CUDB Nodes.

2.1.1 Checking 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 check which SC is the active one:

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

The expected output must be similar to the below example:

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

2.1.2 CUDB Logchecker

CUDB offers a troubleshooting tool called Logchecker, a set of scripts that aim to improve the in-service performance of CUDB, and also help troubleshooting by means of log collection and automatic log analysis.

The CUDB Logchecker consists of the following two scripts:

- `cudbGetLogs`

  This script collects preventive maintenance logs for log analysis. Log collection takes approximately 4-5 minutes, and the resulting log is saved in the following folder:

  `/home/cudb/monitoring/preventiveMaintenance/`

  The name of the log file contains the node ID and a timestamp in the following format:

  <CUDB_node_id>`_`*<YYYYmmddHHMM>*`.log`

  For example, a log file for a CUDB node with ID 99 looks as follows:

```
CUDB_99_201304090025.log
```

For more details about the command, refer to *CUDB Node Commands and Parameters*, Reference [6].

- `cudbAnalyser`

  This script analyzes the logs collected by `cudbGetLogs`.

Note: The `cudbGetLogs` and `cudbAnalyser` scripts can be executed only on active SCs. See Section 2.1.1 on page 3 for more information on how to check the active SC.

By default, Logchecker performs scheduled log analysis at 00:50 and 12:50, and saves the detailed result in the following location:

```
/home/cudb/monitoring/preventiveMaintenance/cron_analys
is.<SC_NAME>.log
```

In the above path, the *<SC_NAME>* variable can be SC_2_1 or SC_2_2.

The automatic log analysis raises or clears alarms according to the analysis result. The severity of the alarms depends on the severity of the detected faults. For further information, refer to *Preventive Maintenance, Logchecker Found Error(s)*, Reference [7].

For more information about the CUDB Logchecker, refer to *CUDB Logchecker*, Reference [8] and *CUDB Node Commands and Parameters*, Reference [6].

2.1.2.1 Manual Log Collection

The CUDB Logchecker logs can be collected manually with the `cudbGetLogs` command. An example output of the command is shown in Example 1.

```
CUDB107 SC_2_1# cudbGetLogs
Starting /opt/ericsson/cudb/OAM/bin/cudbGetLogs ...
Grepping logs and creating /home/cudb/ \
monitoring/preventiveMaintenance/ \
CUDB_107_201304091136.log ...
The log file is saved as : /home/cudb/\
monitoring/preventiveMaintenance/ \
CUDB_107_201304091136.log
CUDB107 SC_2_1#
```

Example 1 Manual Log Collection

Use this content to provide fault information to the next level of Ericsson support.

2.1.2.2 Log Analysis

Two options are available to trigger unscheduled log analysis:

- **cudbAnalyser -a**: This option is used to check the last two log files automatically.
- **cudbAnalyser --logfile *<newer_logfile>* --previous-logfile *<older_logfile>***: This option is used to check and compare two log files defined with the ***<newer_logfile>*** and ***<older_logfile>*** parameters.

  Note: The timestamp of ***<newer_logfile>*** must be newer than that of ***<older_logfile>*** in order to perform a correct analysis.

  Below is an example of how to compare two log files:

  ```
  cudbAnalyser --logfile /home/cudb/monitoring/preventive
  Maintenance/CUDB_28_201304090300.log --previous-logfile
  /home/cudb/monitoring/preventiveMaintenance/CUDB_28_2
  01304080300.log
  ```

2.1.3 CUDB Logs Collection

The `cudbCollectInfo` command creates a compressed archive from the CUDB logs. Use the output of this command to provide fault information to the next level of Ericsson support.

For further information about this command, execute **cudbCollectInfo -h**, or refer to *CUDB Node Commands and Parameters*, Reference [6].

Note: `cudbCollectInfo` is a system-level command, executed on all nodes automatically. Therefore, do not execute it separately on each node.

The command can take from 5 to 20 minutes, depending of the amount of information to be collected.

2.1.4 LDE Status

The status of the Linux Distribution Extension (LDE) platform can be checked with the `cudbHaState` command. To use this command, at least one SC must be in ACTIVE state.

Execute the command on an SC as follows (in the example below, `SC_2_1` is used):

```
SC_2_1# cudbHaState
```

Execute the following command to check the alarm status on all clusters:

```
SC_2_1# cluster alarm --status --all
```

Execute the following command to check the CPU utilization of the blades or VMs:

```
SC_2_1# cudbMpstat
```

The command provides an output in tabular format.

For more details about the commands, refer to *CUDB Node Commands and Parameters*, Reference [6] and *Data Collection Guideline for CUDB*, Reference [9].

2.1.5 ESA Processes

The status of the ESA processes can be checked by performing the following steps:

1. Check that the ESA agents are running on both SC. Execute the following command on both SCs:

   esa status

   The expected output must look similar to the following:

   ```
   [info] ESA Sub Agent is running.
   [info] ESA Master Agent is running.
   [info] ESA PM Agent is running.
   ```

2. Check the ESA cluster status with the following command on any of the SC blades:

   esaclusterstatus

   The expected output must look similar to the example below. One SC blade must be in `M` state, while the other in `(M)` state:

   ```
   M * OAM1 192.168.0.1
   (M) OAM2 192.168.0.2
   ```

   The description of the different states are:

   M ESA Master is located on that SC.

   (M) ESA Slave is located on that SC.

   ***** The SC from where the command was sent.

2.1.6 BC Cluster Status

In addition to the `cudbSystemStatus` printout, check the BC cluster state with the following command on each node:

`SC_2_1#` **/opt/ericsson/cudb/sm_bcclient/bin/BCClient.sh**

An example output is provided below with a short explanation for each part.

The first part of the command output reports the CS process state. Each Data Store (DS) Unit must have two processes running, one in active state, the other in standby state.

```
cs
N140-D0-I1 --> standby
N140-D0-I2 --> active
N140-D1-I1 --> standby
N140-D1-I2 --> active
N140-D5-I1 --> standby
N140-D5-I2 --> active
```

The next part of the output lists the Date Store Unit Groups (DSGs) and the Processing Layer Database (PLDB) of the system. All DSGs and the PLDB (showed as D0) must be listed here and reported as `alive`, followed by the memory usage indicated between square brackets (`[ ]`).

```
dsg
N140-D0 --> alive [10]
N140-D1 --> alive [80]
N140-D5 --> alive [80]
```

Note: In Hybrid Systems where some DSGs consist of different hardware platforms (GEP3 and GEP5), memory usage values are recalculated for DSGs located on GEP5 nodes. The memory levels for GEP5 DSGs shown here are different from those reported by the database cluster Management Client (`ndb_mgm`).

The next part of the output lists if any ‘‘Automatic Handling of Network Isolation’’ task is running for any PLDB or DSG.

```
ahsi
 N65-D0 --> notRunning
 N65-D132 --> notRunning
 N65-D241 --> notRunning
 N65-D253 --> notRunning
```

The next part of the output lists if any ‘‘Self-Ordered Backup and Restore’’ task is running for any PLDB or DSG.

```
sobr
  N185-D0        -->  notRunning
  N185-D1        -->  notRunning
  N185-D5        -->  notRunning
```

The next part of the output lists the SM status report, otherwise known as the System DS List (SDL). In a healthy CUDB system, all sites must report all DSGs hosted there as alive, as seen below.

```
dsgStatusList
```

```
D0 --> [size:46] {S1-N139-D0=alive [10], S2-N140-D0=alive
[10]}
D1 --> [size:46] {S1-N139-D1=alive [80], S2-N140-D1=alive
[80]}
D5 --> [size:46] {S1-N139-D5=alive [80], S2-N140-D5=alive
[80]}
```

The next part of the output shows the SM masterlist report, otherwise known as the Active DS List (ADL). All DSGs and the PLDB must be listed here with the node where it acts as master, as seen below. The numbers inside the first square bracket of each line indicate the timestamp when the listed replica was appointed to master. The numbers in the second square bracket of each line indicate the node on which the master is located, the node on which the former master was located, and the timestamp of the last time when the data was consistent. See the below example on how these values can appear in the second square bracket:

```
[M1-L2-1428447305368]
```

In the above example, `M1` indicates that the master is currently on Node 1. `L2` shows that the former master was on Node 2. Finally, `1428447305368` is the timestamp of the last time when the data was consistent. If the timestamp is 0, then no previous master is remembered.

See the below example for a complete masterlist report:

```
masterList
D0 --> master-S1-N85-D0 [1465893521110] [M85-L176-0]
D1 --> master-S2-N176-D1 [1467879754100] [M176-L85-14678
79714475]
D2 --> master-S1-N85-D2 [1463759511818] [M85-L176-146375
9493298;M176-L176-0]
```

In case of a split brain situation, the BC cluster entry contains information for both split partitions as shown below:

- Information on the current and former master of partition one: `M85-L176-1463759493298`
- Information on the current and former master for partition two: `M176-L176-0`

The next part of the output lists the SM partition status report. In a healthy system, it must report majority status, as seen below. `AR` means ‘‘Auto Removed’’ sites in case any site went down. `NS` states ‘‘Not Started’’ sites, when there are connectivity to the site, but the BC Cluster is down.

```
status --> majority[1, 2] AR[] NS[]
```

The next part of the output contains the SM leader report, showing the SM site leader instance:

```
leader --> S2-N140-I2
```

The next part of the output contains the remote SM connectivity report, showing the remote node instances:

```
auxiliar
S1-N139-I1 --> Hello
S1-N139-I2 --> Hello
```

The next part of the output shows the System Working Mode (mainly used for upgrades):

```
working
mode --> normal
```

The next part of the output shows the list of ongoing and pending transfers:

```
transferQueue
```

The next part of the output shows the action given to the System Monitor:

```
actions
 masterMovement
```

The last part of the output shows the alarms raised and ready to be cleared:

```
alarms
 clear
 raise
```

Geographical data redundancy in CUDB is achieved by asynchronous data replication from the master replicas of each DSG (or the PLDB) to the corresponding slave replicas. Changes in the data stored in the master replica typically take a short time to be replicated to the slave replicas. The time between the moment a piece of data is changed in the master replica, and the moment that change is applied on the slave replica is called ‘‘replication lag’’.

If the command is executed with the `-tree | -T` option, additional information can be obtained, like the replication status mentioned above. In such cases, the output follows the below syntaxes:

- If the DSG is up, the output follows the below format:

  <nodeID>-*<dsgId>* --> *<replication_status>*[*<active_channel_number>*, *<number_of_pending_changes>*, *<replication_lag_delay>*]

- If the DSG is down (for example because of maintenance), then the output follows the below format:

  <nodeID>-*<dsgId>* --> down[*<timestamp>*]

The contents of the above syntax variations are as follows:

- *<nodeID>* is the Node ID.
- <dsgId> is the DSG IDs.
- <replication_status> indicates if replication is working or not. Its value is either up or down.
- <active_channel_number> indicates the number of the active replication channel. Its value is either 1 or 2.
- *<number_of_pending_changes>* indicates the number of pending changes.
- *<replication_lag_delay>* indicates the replication lag in seconds.
- *<timestamp>* indicates the timestamp.

See the below output for an example of both syntaxes:

```
N161-D0 --> up[1, 353, 0.1]
 N161-D1 --> down[1470214420]
 N161-D255 --> up[2, 201, 0.1]
```

2.1.7 Check Database Consistency between Database Clusters

Use the cudbCheckConsistency command to check database consistency in a lightweight manner between database clusters (that is, between the master PLDB or DSG replicas and their slaves) as follows:

```
SC_2_1# cudbCheckConsistency
```

For more details about the command, refer to *CUDB Node Commands and Parameters*, Reference [6].

2.1.8 cudbConsistencyMgr

Use the cudbConsistencyMgr command to check database consistency between database clusters (that is, between the master PLDB or DSG replicas and their slaves) as follows:

```
SC_2_1# cudbConsistencyMgr --order ms --node <nodeid>
{--dsg <dsgid> | --pl}
```

For more details about the command, refer to *CUDB Node Commands and Parameters*, Reference [6].

2.1.9 Check Replication Channels Behavior

Replication channels on CUDB system level can be checked by generating dummy transactions, and then verifying that those transactions are replicated properly for each DSG and PLDB slave replica.

Use the `cudbCheckReplication` command as follows to generate dummy transactions:

```
SC_2_1# cudbCheckReplication
```

For more details about the command, refer to *CUDB Node Commands and Parameters*, Reference [6].

2.2 UDC Cockpit Tool

To follow present and recall earlier system status and performance information of CUDB nodes, use the UDC Cockpit. This is a monitoring application, which presents collected data on a single, web-based GUI.

3 Troubleshooting Procedure

A troubleshooting workflow is shown in Figure 1.

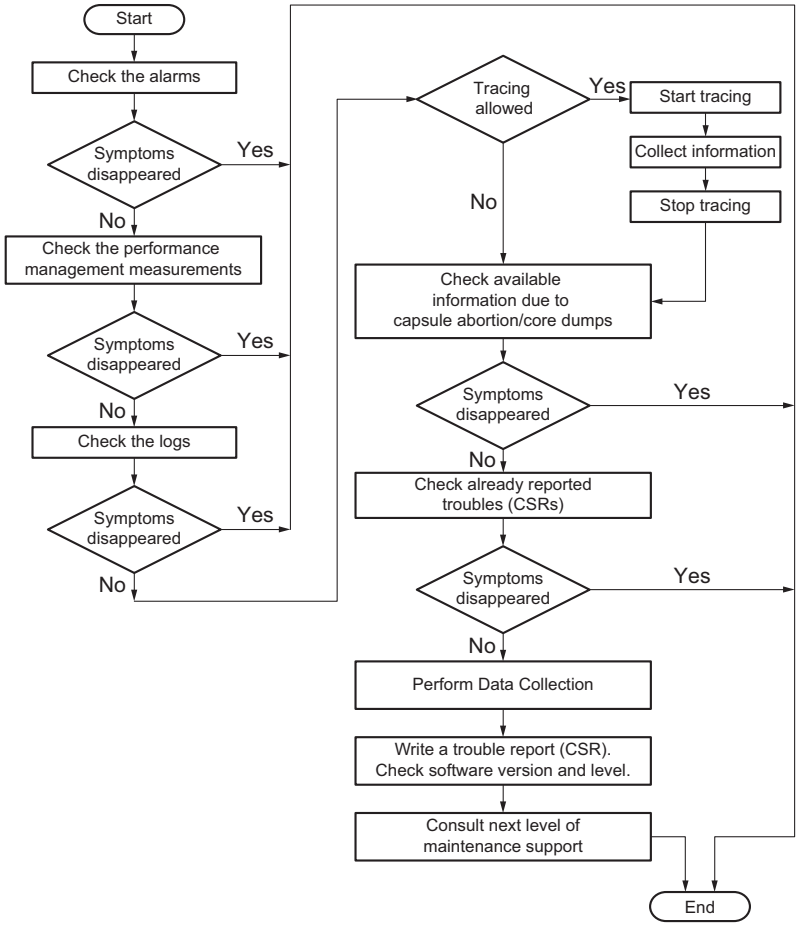

Figure 1 Troubleshooting Workflow

4 Trouble Reporting

Problems identified that cannot be solved by using this document must be reported to the next level of maintenance support through a Customer Service Report (CSR).

The details of the trouble reporting process is outside the scope of this document.

When collecting information for further support, ensure that all current logs are recorded. See time and date for the logs.

For more information on how to collect information, refer to *Data Collection Guideline for CUDB*, Reference [9].

When sending crash dumps, ensure that the dump is of the actual scenario. See time and date for the dump.

Glossary

For the terms, definitions, acronyms and abbreviations used in this document, refer to *CUDB Glossary of Terms and Acronyms*, Reference [3].

Reference List

CUDB Documents

[1] *CUDB Node Preventive Maintenance*

[2] *CUDB Users and Passwords*, 3/00651-HDA 104 03/10

[3] *CUDB Glossary of Terms and Acronyms*

[4] *Trademark Information*

[5] *Typographic Conventions*

[6] *CUDB Node Commands and Parameters*

[7] *Preventive Maintenance, Logchecker Found Error(s)*

[8] *CUDB Logchecker*

[9] *Data Collection Guideline for CUDB*